受領書

平成１１年　９月１７日
特　許　庁　長　官

識別番号　　１００１０３６３２
氏名（名称）　　窪田　英一郎　　殿
提出日　　平成１１年　９月１７日

以下の書類を受領しました。

項番	書類名	整理番号	受付番号	出願番号通知（事件の表示）
1	特許願	PK990016	59900905723	特願平11-263793

以　上

【書類名】　特許願

【整理番号】　PK990016

【あて先】　特許庁長官　　殿

【国際特許分類】　G06F 15/16

G06F 15/80

【発明者】

【住所又は居所】　神奈川県横浜市神奈川区松見町４丁目１１０１番地７コートハウス菊名８０４号

【氏名】　古庄　晋二

【特許出願人】

【住所又は居所】　東京都台東区松が谷１－９－１２　ＳＰＫビルディング６０４号

【氏名又は名称】　ターボデータラボラトリー有限会社

【代表者】　古庄　晋二

【代理人】

【識別番号】　100103632

【弁理士】

【氏名又は名称】　窪田　英一郎

【選任した代理人】

【識別番号】　100099715

【弁理士】

【氏名又は名称】　吉田　聡

【手数料の表示】

【予納台帳番号】　058377

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　並列コンピュータのアーキテクチャおよびこのアーキテクチャを利用した情報処理ユニット

【特許請求の範囲】

【請求項１】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された並列コンピュータのアーキテクチャであって、

所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、

各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする並列コンピュータのアーキテクチャ。

【請求項２】　前記ＭＰＵが、ＣＰＵから与えられた空間ＩＤを、自己が管理する１以上の一連のデータの空間ＩＤと比較する空間コンパレータと、

ＣＰＵから与えられた論理アドレスと、自己が管理するデータの部分の論理アドレスとを比較するアドレスコンパレータと、

当該論理アドレスに基づき、自己のＲＡＭセル上の物理アドレスを算出するアドレスカリキュレータとを有することを特徴とする請求項１に記載のコンピュータアーキテクチャ。

【請求項３】　前記メモリモジュールの各々が、ＣＰＵモジュールおよび他のメモリモジュールとの同期をなすための同期信号を受け入れ、かつ、

前記複数組のバスの何れかとの接続が可能な入力と、前記複数組のバスの他の何れかとの接続が可能な出力を備え、少なくとも、前記同期信号にしたがって、前記何れかのバスと入力との接続により、データを入力しつつ、前記他の何れかのバスと出力との接続により、データを出力できるように構成されたことを特徴とする請求項１または２に記載のコンピュータアーキテクチャ。

【請求項４】　前記複数組のバスの各々に、前記ＣＰＵモジュールと何れかのメモリモジュールの入力または出力との間、および／または、他の何れかのメモリモジュールの入力または出力と、さらに他のメモリモジュールの出力または入力との間の接続を規定するためのスイッチが設けられ、

前記スイッチの切換により、複数組のバスの各々において、並列的にデータの授受が実現されることを特徴とする請求項３に記載のコンピュータアーキテクチャ。

【請求項５】　前記複数組のバスのうちの何れかである第１のバスに、何れかのメモリモジュールの出力と、他の何れかのメモリモジュールの入力とが接続され、かつ、前記複数組のバスのうち、他の何れかである第２のバスに、当該他の何れかのメモリモジュールの出力と、さらに他の何れかのメモリモジュールの入力とが接続され、第１のバスにおけるデータの授受と、第２のバスにおけるデータの授受が並列的に進行することを特徴とする請求項４に記載のコンピュータアーキテクチャ。

【請求項６】　前記バスとメモリモジュールとの間の接続を繰り返して、多段のメモリモジュール間の接続を形成することを特徴とする請求項５に記載のコンピュータアーキテクチャ。

【請求項７】　前記ＭＰＵが、一連のデータ中の特定の要素を削除し、前記一連のデータ中に特定の要素を挿入し、或いは、一連のデータの末尾に特定の要素を追加することを示すインストラクションを受理すると、テーブルを参照して、自己の管理するデータの領域と、削除、挿入或いは追加にかかる要素の位置とを比較して、当該比較結果に応じて、前記テーブルの内容を更新することを特徴とする請求項１ないし６の何れか一項に記載のコンピュータアーキテクチャ。

【請求項８】　前記ＭＰＵが、与えられたインストラクションに応答して、一

連のデータ中の要素を特定するための添え字を変換し、および／または、要素に特定の修飾を与える値変換を実行することを特徴とする請求項１ないし７の何れか一項に記載のコンピュータアーキテクチャ。

【請求項９】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された情報処理ユニットであって、

所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、

各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする情報処理ユニット。

【請求項１０】　前記ＣＰＵモジュールが、レガシーメモリ、入力装置および表示装置を相互接続する他のバスと連結可能に構成されたことを特徴とする請求項９に記載の情報処理ユニット。

【請求項１１】　請求項９に記載の情報処理ユニットと、ＣＰＵモジュールと他のバスを介して連結された１以上のレガシーメモリを含む記憶装置、入力装置および表示装置とを有することを特徴とするコンピュータシステム。

【発明の詳細な説明】

【０００１】

【産業上の技術分野】

本発明は、ＳＩＭＤ(Single Instruction Stream, Multiple Data Stream)を実現可能な並列コンピュータのアーキテクチャに関し、より詳細には、適切かつ

高速なメモリ制御により、汎用的な並列演算が可能なコンピュータアーキテクチャに関する。

【０００２】

【従来の技術】

社会全体のさまざまな場所にコンピュータが導入され、インターネットをはじめとするネットワークが浸透した今日では、そこここで、大規模なデータが蓄積されるようになった。このような大規模データを処理するには、膨大な計算が必要で、そのために並列処理を導入しようと試みるのは自然である。

【０００３】

さて、並列処理アーキテクチャは「共有メモリ型」と「分散メモリ型」に大別される。前者（「共有メモリ型」）は、複数のプロセッサが１つの巨大なメモリ空間を共有する方式である。この方式では、プロセッサ群と共有メモリ間のトラフィックがボトルネックとなるので、百を越えるプロセッサを用いて現実的なシステムを構築することは容易ではない。したがって、例えば１０億個の浮動小数点変数の平方根を計算する際、単一ＣＰＵに対する加速比は、せいぜい１００倍ということになる。経験的には、３０倍程度が上限である。

後者（「分散メモリ型」）は、各プロセッサがそれぞれローカルなメモリを持ち、これらを結合してシステムを構築する。この方式では、数百～数万ものプロセッサを組み込んだハードウェアシステムの設計が可能である。したがって、上記１０億個の浮動小数点変数の平方根を計算する際の単一ＣＰＵに対する加速比を、数百～数万倍とすることが可能である。しかしながら、後者においても、後述するいくつかの課題が存在する。

本出願は、「分散メモリ型」に関するものであり、この方式について最初に多少の考察を加えながら従来技術との比較を行うことにする。

【０００４】

【課題を解決するための手段】

[第１の課題：巨大配列の分掌管理]

「分散メモリ型」の第１の課題は、データの分掌管理の問題である。

巨大なデータ（一般的には配列なので、以降、配列で説明する）は、１つのプ

ロセッサの所有するローカルメモリに収容できるものではなく、必然的に複数のローカルメモリに分掌管理される。効率的かつ柔軟な分掌管理メカニズムを導入しないと、プログラムの開発および実行に際してさまざまな障害を抱え込むことになることは明らかである。

【０００５】

[第２の課題：プロセッサ間通信の効率の低さ]

分散メモリ型システムの各プロセッサが、巨大配列にアクセスしようとすると、自己の所有するローカルメモリ上の配列要素に対しては速やかにアクセスできるものの、他のプロセッサが所有する配列要素へのアクセスはプロセッサ間通信を必須とする。このプロセッサ間通信はローカルメモリとの通信に比べ、極端にパフォーマンスが低く、最低でも１００クロックかかると言われている。このため、ソート実施時には、巨大配列全域にわたる参照が実施され、プロセッサ間通信が多発するため、パフォーマンスが極端に低下する。

【０００６】

この問題点につき、より具体的に説明を加える。１９９９年現在、パソコンは、１～数個のＣＰＵを用いて、「共有メモリ型」として構成されている。このパソコンに使用される標準的なＣＰＵは、メモリバスの５～６倍程度の内部クロックで動作し、その内部に自動的な並列実行機能やパイプライン処理機能が装備されており、およそ１データを１クロック（メモリバス）で処理できる。

「共有メモリ型」であるパソコンにて巨大配列のソート処理を行う場合、１データについて１クロックを要し、このため、１データに１００クロック（メモリバス）を要する、「分散メモリ型」のマルチプロセッサシステムの１００倍のパフォーマンスを発揮することも考えられる。

【０００７】

[第３の課題：プログラムの供給]

「分散メモリ型」の第３の課題は、多数のプロセッサにどうやってプログラムを供給するか、という問題である。

非常に多数のプロセッサに、別々のプログラムをロードし、全体を協調動作させる方式（ＭＩＭＤ：Multiple Instruction Stream, Multiple Data Stream）

では、プログラムの作成、コンパイル、配信のために多大な負荷を要する。

その一方、多数のプロセッサを同一のプログラムで動作させる方式（ＳＩＭＤ：Single Instruction Stream, Multiple Data Stream）では、プログラムの自由度が減少し、所望の結果をもたらすプログラムが開発できない事態も想定される。

【０００８】

本発明は、「分散メモリ型」の上記第１ないし３の課題を解決する方法およびコンピュータアーキテクチャを提供する。第１の「巨大配列の分掌管理」の課題は、配列の各要素の配置（物理アドレス）を、各プロセッサモジュールが統一的な方法で分掌管理することで解決できる。この手法により、ガーベージコレクションの必要性が無くなり、配列要素の挿入・削除が数クロックで完了し、ＳＩＭＤを実現する上で欠かせない各プロセッサの暗黙の（非明示的）処理分担を割り付けることもできる。この方法は、後ほど「多空間メモリ」という概念で説明される。

【０００９】

第２の「プロセッサ間通信の効率の低さ」の課題は、達成しようとする処理に応じて各プロセッサ間をつなぎ替え、各接続経路毎に、定められた種類のデータを、定められた順番で、１方向に連続転送することで、バスの能力を１００％近くまで使用できるよう通信をスケジュール化し、同時に巨大パイプライン処理を実現することで解決できる。

その有効性を実証するため、後ほど、現実的なシステム設計で、１０億行のソートを１秒程度で完了するシステムの構成方法を例示するであろう。これは、既知の最高速の装置に比べて、１万倍以上高速である。この方法は、後ほど「組替えバス」技術として説明される。

【００１０】

第３の「プログラムの供給」の課題は、ＳＩＭＤ方式を採用することで解決できる。ＳＩＭＤの場合は、各プロセッサの暗黙の（非明示的）処理分担をどうやって決定するか？が最大の問題であるが、前述の「多空間メモリ」技術にてこの処理分担が自動的に決定でき、ＳＩＭＤであってもプログラムの自由度を保持す

ることができる。

つまり、本発明は、分散メモリー型において、単一命令により種々のメモリーに記憶された配列中の要素を入出力し、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供することを目的とする。

【００１１】

【課題を解決するための手段】

本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された並列コンピュータのアーキテクチャであって、所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする並列コンピュータのアーキテクチャにより達成される。

【００１２】

本発明によれば、空間ＩＤを用いて一連のデータを把握するため、当該一連のデータが、多数のメモリモジュールにより分掌されても、各メモリモジュールのＭＰＵが、当該一連のデータを確実に認識することができる。また、メモリモジュールは、一連のデータおよび自己が管理するその部分を、テーブルにて把握しているため、インストラクションの受理にしたがって、そのテーブルを参照して、所定の処理を実行することができる。これにより、単一インストラクションに基づく、各ＭＰＵでの並列処理が実現できる。

【００１３】

本発明の好ましい実施態様においては、ＭＰＵは、ＣＰＵから与えられた空間ＩＤを、自己が管理する１以上の一連のデータの空間ＩＤと比較する空間コンパレータと、ＣＰＵから与えられた論理アドレスと、自己が管理するデータの部分の論理アドレスとを比較するアドレスコンパレータと、当該論理アドレスに基づき、自己のＲＡＭセル上の物理アドレスを算出するアドレスカリキュレータとを有している。これらコンパレータおよびカリキュレータは、ハードウェアにて構成されても良いし、ＭＰＵのプログラムによりソフトウェアとして実現されるものであっても良い。

【００１４】

また、本発明の好ましい実施態様においては、メモリモジュールの各々が、ＣＰＵモジュールおよび他のメモリモジュールとの同期をなすための同期信号を受け入れ、かつ、前記複数組のバスの何れかとの接続が可能な入力と、前記複数組のバスの他の何れかとの接続が可能な出力を備え、少なくとも、前記同期信号にしたがって、前記何れかのバスと入力との接続により、データを入力しつつ、前記他の何れかのバスと出力との接続により、データを出力できるように構成されている。

本実施の形態によれば、同期信号にしたがって、メモリモジュールからのデータ出力およびメモリモジュールへのデータ入力がなされ、かつ、バスの接続の制御により、適切に並列処理を実現することが可能となる。

【００１５】

複数組のバスの各々には、前記ＣＰＵモジュールと何れかのメモリモジュールの入力または出力との間、および／または、他の何れかのメモリモジュールの入力または出力と、さらに他のメモリモジュールの出力または入力との間の接続を規定するためのスイッチが設けられ、スイッチの切換により、複数組のバスの各々において、並列的にデータの授受が実現されるのがより好ましい。これにより、複数組のバスをより有効に利用することが可能となり、より並列性を高めることが可能となる。

【００１６】

本発明のさらに好ましい実施態様においては、複数組のバスのうちの何れかで

ある第１のバスに、何れかのメモリモジュールの出力と、他の何れかのメモリモジュールの入力とが接続され、かつ、前記複数組のバスのうち、他の何れかである第２のバスに、当該他の何れかのメモリモジュールの出力と、さらに他の何れかのメモリモジュールの入力とが接続され、第１のバスにおけるデータの授受と、第２のバスにおけるデータの授受が並列的に進行する。このように、コンピュータの実施態様によれば、ＣＰＵモジュールと、メモリモジュールとにより、パイプライン処理を実現することが可能となる。バスとメモリモジュールとの間の接続を繰り返して、多段のメモリモジュール間の接続を形成するのがより好ましい。

【００１７】

本発明の別の好ましい実施態様においては、ＭＰＵが、一連のデータ中の特定の要素を削除し、前記一連のデータ中に特定の要素を挿入し、或いは、一連のデータの末尾に特定の要素を追加することを示すインストラクションを受理すると、テーブルを参照して、自己の管理するデータの領域と、削除、挿入或いは追加にかかる要素の位置とを比較して、当該比較結果に応じて、前記テーブルの内容を更新する。すなわち、ＭＰＵにおいて、自己が管理するテーブルを更新する、すなわち、リマッピングをすることにより、要素の削除、挿入および追加を実現することが可能となる。

【００１８】

本発明のさらに別の実施態様においては、ＭＰＵが、与えられたインストラクションに応答して、一連のデータ中の要素を特定するための添え字を変換し、および／または、要素に特定の修飾を与える値変換を実行する。

また、本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された情報処理ユニットであって、所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデー

タの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする情報処理ユニットによっても達成される。たとえば、前記ユニットが単一の回路基板に形成され、ＣＰＵモジュールが、レガシーメモリ、入力装置および表示装置を相互接続する他のバスと連結可能に構成されていても良い。

【００１９】

さらに、本発明の目的は、上記情報処理ユニットと、ＣＰＵモジュールと他のバスを介して連結された１以上のレガシーメモリを含む記憶装置、入力装置および表示装置とを有することを特徴とするコンピュータシステムによっても達成される。

【００２０】

【発明の実施の形態】

[ハードウェア構成]

以下、添付図面を参照して、本発明の実施の形態につき説明を加える。図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。図１に示すように、コンピュータシステム１０は、単一命令による並列演算を実現するＣＰＵモジュール１２と、並列演算のために必要な種々のデータを記憶するメモリモジュール１４－１、１４－２、１４－３、…と、必要なプログラムやデータを記憶する固定記憶装置１６と、キーボードやマウスなどの入力装置１８と、ＣＲＴなどからなる表示装置２０と、種々の形式のデータ等が記憶されているレガシーメモリ２２とを備えている。また、バス２４－１、２４－２、…において、ＣＰＵモジュール１２、各メモリモジュール１４との接点には、スイッチ２８－１、２８－２、２８－３、…などが配設され、選択された回路要素間における情報の授受が可能となっている。また、ＣＰＵモジュール１２とメモリモジュール１４－１との間、隣接するメモリモジュール間において

、バスの連結および接続をなすためのスイッチ３０－１、３０－２、…が設けられている。

【００２１】

ＣＰＵモジュール１２と、メモリモジュール１４との間には、複数のバス２４－１、２４－２、２４－３、２４－４、…とが設けられている。したがって、ＣＰＵモジュール１２とメモリモジュール１４との間、および、メモリモジュール間は、上記バスによりデータ等の授受が可能となっている。また、ＣＰＵ１２と、メモリモジュール１４との間には、制御信号ライン２５が設けられ、ＣＰＵ１２から発せられるインストラクションなどが、全てのメモリモジュール１４に伝達されるようになっている。

【００２２】

さらに、ＣＰＵ１２と、他の構成要素（たとえば、固定記憶装置１６、入力装置１８など）との間には、ローカルバス２６が配設されており、これらの間でもデータ等の授受が可能となっている。ＣＰＵ１２は、固定記憶装置１６に記憶され、或いは、バス２６上に接続されたＲＡＭのような他の記憶装置（図示せず）に記憶されたプログラムを読み出し、このプログラムにしたがって、以下に示すメモリモジュール１４へのインストラクションの送出を含むデータの授受のほか、スイッチ２８、３０の制御等を実行する。また、ＣＰＵ１２は、プログラムにしたがって、レガシーメモリ２２に記憶された種々の形式のデータを受け入れて、この形式のデータを、ＣＰＵ１２、メモリモジュール１４、バス２４からなる系にて処理可能な一連のデータ（配列）に変換し、これらを、各メモリモジュール１４に記憶させることもできる。

【００２３】

図２は、各メモリモジュール１４の概略を示すブロックダイヤグラムである。図２に示すように、メモリモジュール１４は、ＣＰＵモジュール１２から与えられるクロックなど同期信号を受け入れるクロックバッファ３２と、データを記憶するＲＡＭコア３４と、後述する空間ＩＤやデータの要素番号等を把握し、ＣＰＵ１２からのインストラクションなどを受理した場合に、空間ＩＤや要素番号に基づき、ＲＡＭコア３４へのデータ書き込みやＲＡＭコアからのデータ読み出し

を制御するＭＰＵ３６と、バスの何れかからのデータを受け入れて、ＲＡＭコア３４に供給し、および／または、ＲＡＭコア３４からのデータを何れかのバスに送出するＩ／Ｏ３８とを有している。この実施の形態において、メモリモジュール１４は、制御信号ライン２５を介して、ＣＰＵからのインストラクションを受け入れ、ＭＰＵ３６が、このインストラクションに応答して、ＲＡＭコア３４のデータを読み出し、ＲＡＭコア３４にデータを書き込み、或いは、データに所定の処理を施すことができるようになっている。また、ＲＡＭコア３４へのデータアクセスや、Ｉ／Ｏを介してデータ入力およびデータ出力は、クロックバッファ３２に与えられるクロックなどの同期信号に基づき実行される。

【００２４】

図１および図２から明らかなように、本発明において、コンピュータシステム１０は、メモリ共有型のシステムであると考えることができる。また、後述するように、制御信号ライン２５を介して、各メモリモジュール１４にインストラクションを与えることにより、各メモリモジュール１４が並列的に処理を実行する。また、バスへのデータ出力およびバスからのデータ入力などが、所定の同期信号に基づき実行される。したがって、このコンピュータシステム１０は、ＳＩＭＤの形態をなしていると考えることができる。

【００２５】

[実現される機能の概略]

このような構成を有するコンピュータシステム１０につきより詳細な説明を加える前に、本コンピュータシステム１０により実現される機能の概略を簡単に説明する。

（１）多空間メモリ

本明細書において、多空間メモリとは、メモリ空間を、空間ＩＤとアドレスとに基づきアクセスするために割り当てられたメモリ空間をいう。これにより、一連のデータが多数のプロセッサに分掌されていても、各プロセッサが、これを確実に分離、認識することができる。

従来のメモリ空間においては、プロセス毎に個別の領域を割り当てることはあっても、一連の変数（配列、構造体など）毎に目盛り空間を割り当てることは行

われてこなかった。したがって、以下、このような従来のメモリ空間を「単一メモリ空間」と称する。単一メモリ空間のシステムにおいては、アドレスのみを用いてデータにアクセスしているため、関連を有する一連のデータを分離したり、認識することができなかった。このため、実際には並列処理が可能であっても、その可否を判断できない場合が多かった。また、ある単一メモリ空間に、新たな一連のデータを収容させる場合に、当該一連のデータの収容場所を確保するために、ガーベージコレクションを実行する必要があった。

【００２６】

これに対して、本発明においては、メモリ空間に、空間ＩＤを導入し、一連のデータについて同一のＩＤを付与している。また、メモリモジュール１４において、自身のＲＡＭコア３４に保持されているデータに関する空間ＩＤを把握し、これにより、各メモリモジュール１４自体が、現在アクセスされているデータの空間ＩＤを参照することにより、自己の作動の是非を決定することができる。また、各メモリモジュールが空間ＩＤと関連付けて、一連のデータの全部或いは一部を保持できるため、ある一連のデータを、複数のメモリモジュール１４に分割して記憶させることができ、これによりガーベージコレクションを不要にすることができる。

【００２７】

たとえば、図３に示すように、単一メモリ空間において、“Ａ”という一連のデータ、“Ｂ”という一連のデータ、…が収容されている場合を考える。たとえば、ここで、全メモリサイズが３２ワードで、上記一連のデータのサイズの総和が３０ワードであると仮定する。これら一連のデータは、空間中に点在しているため、未使用のメモリサイズは、１２ワードであるにもかかわらず、実際に格納できる一連のデータのサイズは３ワードに限定される。このため、３ワードを超えたサイズを有する新たな一連のデータを収容すべき場合には、ガーベージコレクションを実行しなければならない。その一方、図４に示すように、本発明においては、一連のデータの各々に、空間ＩＤが付与されている。これらは、空間ＩＤと関連付けられて、１以上のメモリモジュール１４に記憶される。したがって、未使用のサイズと収容可能なサイズとを一致させることが可能となる。

【００２８】

（２）メモリモジュール

また、本発明においては、各メモリモジュール１４が、ＭＰＵ３６を有し、上記空間ＩＤのほか、自己が保持する一連のデータの各々の要素番号を把握している。したがって、ＣＰＵ１２からのインストラクションを受理した後、ＭＰＵ３６が、インストラクションにしたがってアクセスすべきデータが、自己のＲＡＭコア３４中に保持されているものか否かを判断して、アクセスに必要の是非を決定することができる。さらに、各メモリモジュール１４が、自己のＲＡＭコア３４に格納されている配列要素の添え字の範囲から、ＳＩＭＤでのインストラクションにおける暗黙の処理の分担範囲を決定することが可能である。

【００２９】

また、本発明においては、メモリモジュール１４が、アドレスリマッピングを実行できるようになっている。たとえば、図５に示すように、ある配列の所定の位置に特定の要素を挿入する場合、その他、所定の位置の要素を削除し、或いは、配列の末尾に所定の要素を追加する場合にも、本実施の形態においては、当該配列に関連する要素を保持しているメモリモジュールの各々において、ＭＰＵ３６が、アドレスリマッピングを実行することにより、並列的かつ高速に、これらを実現することができる。さらに、図６に示すように、配列の要素（値）に修飾を与える場合（たとえば、各値に「１」を加える場合）にも、関連する配列の要素を保持するメモリモジュールの各々において、ＭＰＵ３６が、並列的かつ高速に、必要な処理を行うことができる。

【００３０】

また、メモリモジュール１４においては、ＭＰＵ３６が、ＲＡＭコア３４にて記憶すべきデータの各々のサイズを把握し、圧縮した形態にてこれらを記憶することができる。たとえば、あるメモリモジュール１４にて、整数値のデータを保持すべき場合に、実際のデータ値が“０”ないし“３”までの値しか取り得ない場合には、ＭＰＵ３６は、各データのために２ビットのみを用意する。ＣＰＵ１２との間では、１つの整数を表現するために３２ビットを使用していた場合には、メモリモジュール１４とＣＰＵ１２との間での通信のために、ＭＰＵ３６が、

データ形式を変更して、ＣＰＵ１２との授受をなせば良い。これにより、ＲＡＭコア３４をより無駄なく利用することが可能となる。また、文字列のような長さの異なるデータについても、同様にデータ長を変更して記憶することができるようになっている。

【００３１】

さらに、メモリモジュール１４においては、所定の空間ＩＤに関連付けられたデータや、所定の範囲の要素番号を付されたデータに、特定の値（たとえば、「０」）をセットすることができるようになっている。これにより、メモリモジュール１４内で、高速に初期化の処理を実行することが可能となる。また、メモリモジュール１４においては、ある特定のデータ（配列）中の値を検索することや、添字の範囲をチェックすることが可能である。

【００３２】

（３）組み替え可能バス

本発明においては、ＣＰＵ１２が、スイッチ２８－１、２８－２、…およびスイッチ３０－１、３０－２、…を選択的にオン／オフして、データの授受をなすべきメモリモジュール１４を指定することにより、パイプライン処理を実現している。たとえば、図７に示すように、あるメモリモジュール１４－ｉから出力されたデータを、他のメモリモジュール１４－ｊに与え、かつ、当該他のメモリモジュール１４－ｊから出力されたデータを、さらに他のメモリモジュール１４－ｋに伝達すべき場合には、ＣＰＵ１２は、バス２４－ｍを、メモリモジュール１４－ｉ、１４－ｊのために割り当て、かつ、バス２４－ｎを、メモリモジュール１４－ｊ、１４－ｋのために割り当てるように、各スイッチの状態を設定する。

【００３３】

さらに、これらパイプライン処理は、単一のメモリモジュール間の接続により実現される場合だけでなく、複数の一連のメモリモジュール（メモリモジュール群）の間の接続により実現することも可能である。達成しようとする処理に応じて、各メモリモジュール間をつなぎ替え、各接続経路毎に、定められた種類のデータを定められた順序にて一方向に連続転送することで、バスの能力を１００％近く使用できるように、通信をスケジュール化することができる。これにより、

分散メモリ型の並列処理システムの最大の問題であった、プロセッサ間通信のパフォーマンスの低さを、解消することができる。

このように構成されたコンピュータシステム１０において、多空間メモリの具体的構成および多空間メモリにおけるシステムの作動につき説明を加える。

【００３４】

［多空間メモリ］

図８は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。図８（ａ）に示すように、メモリモジュール１４中のＲＡＭコア３４には、空間ＩＤ管理テーブルが設けられる。これにより、メモリモジュール１４のＭＰＵ３６は、自己が保持するデータの空間ＩＤ等必要な情報を把握することが可能となる。

図８（ｂ）に示すように、空間ＩＤ管理テーブルには、自己が保持するデータ群ごとの、空間ＩＤ、ＣＰＵの管理の下での、データ群の論理開始アドレス、データ群が割り付けられた領域のサイズ、ＲＡＭコア３４中の物理開始アドレス、当該空間ＩＤを有する一連のデータの全サイズ、および、アクセス制限を示すアクセス制限フラグが格納されている。アクセス制限フラグは、この実施の形態においては、読み出しのみ可能（Ｒ）、書き込みのみ可能（Ｒ）、読み書き可能（ＲＷ）の３つの状態を示すことができるようになっている。

【００３５】

メモリモジュール１４のＭＰＵ３６は、ある空間ＩＤを有するデータ群が与えられた際に、ＲＡＭコア３４中に当該データ群を収容すべき、１以上の領域を見出して、当該領域にデータ群をそのまま、或いは、２以上に分割して収容する。この際に、与えられた空間ＩＤ、論理開始アドレス、全サイズ、アクセス制限フラグとともに、実際にデータを収容したＲＡＭコア中の論理開始アドレスや、割り付け領域サイズも、空間ＩＤ管理テーブルに記憶される。図８（ｃ）は、図８（ｂ）による空間ＩＤ管理テーブルにしたがったＲＡＭコア３６中のデータを示す図である。

【００３６】

［メモリアクセスの概略説明］

このように構成されたメモリモジュール１４へのアクセスにつき以下に説明を加える。図９に示すように、まず、ＣＰＵ１２が、空間ＩＤおよび論理アドレス、並びに、必要なインストラクション（たとえば、データの書き込みや読み出し）を、制御信号ライン２５を介して、全てのメモリモジュール１４に伝達する。各メモリモジュール１４においては、これに応答して、ＭＰＵ３６に設けられた空間コンパレータ５２が、空間ＩＤと、自己の空間ＩＤ管理テーブル上に保持されている空間ＩＤとを比較して、同一のものを、自己が保持しているかを判断し、また、、アドレスコンパレータ５４が、論理アドレスについて、同様の判断を行う。次いで、メモリモジュール１４のＭＰＵ３６が、自己のＲＡＭコア３４に、インストラクションによる処理対象となるデータが保持されていると判断した場合には、アドレスカリキュレータ５６が、空間ＩＤ管理テーブルを参照して、ＲＡＭコア３４中の物理アドレスを算出し、処理対象となるデータを特定する。

このようにして、データが特定された後に、ＭＰＵ３６は、ＣＰＵ１２から与えられたインストラクションに応じた処理（たとえば、データの書き込みや読み出し）を実行し、必要な場合には、データをＣＰＵ１２に伝達する（図９（ｃ）参照）。

【００３７】

［多空間メモリのより具体的な動作：配列中の要素の削除等］

たとえば、ある空間ＩＤをもつ一連のデータ（以下、これを場合によって「配列」と称する。）が、１以上のメモリモジュール１４に収容された状態から、特定の要素が削除された状態までの一連の動作につき以下に説明する。

あるメモリモジュール１４－ｉにおいて、空間ＩＤ「０１０」に属するデータ群が、図１０（ａ）に示すように格納され、多のメモリモジュール１４－ｊにおいて、空間ＩＤ「０１０」に属するデータ群が、図１０（ｂ）に示すように格納されている場合を考える。たとえば、メモリモジュール１４－ｉにおいては、論理アドレス「０」から「５９」までのデータが、そのＲＡＭコアの物理アドレス「１００」から記憶されていることがわかる。この場合に、みかけの配列は、図１０（ｃ）に示すようなものとなる。

【００３８】

このように複数のメモリモジュールに、ある配列が格納されている場合に、特定の要素を削除する際の処理につき以下に述べる。ＣＰＵ１２から、各メモリモジュール１４－１、１４－２、…に、制御信号ライン２５を介して、空間ＩＤ「０１０」の要素「５０～５９」を削除するというインストラクションが発せられた場合を考える。図１１および図１３は、ある空間ＩＤ中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【００３９】

各メモリモジュールのＭＰＵ３６は、制御信号ライン２５を介して与えられたインストラクションを受理して、その内容を解釈し（ステップ１１０１）、インストラクション中の「空間ＩＤ」を調べ（ステップ１１０２）、自己のＲＡＭコア３４が保持するデータの空間ＩＤに関連しているか否かを判断する（ステップ１１０３）。ステップ１１０３にてノー（No）と判断された場合には、処理を終了し、その一方、イエス（Yes）と判断された場合には、ＭＰＵ３６は、空間ＩＤ管理テーブルを参照して、当該空間ＩＤに関するデータ群が書き込み可能な状態になっているか、或いは、削除要求のあった範囲のサイズが、全サイズよりも小さいか否かなどを判断する（ステップ１１０４）。チェックによって異常があると判断された場合（ステップ１１０５でイエス(Yes)）には、ＭＰＵ３６は、制御信号ライン２５を介してエラーが生じたことを通知する。その一方、異常がない場合には、ＭＰＵ３６は、インストラクションにより削除を要求された範囲と、自己のＲＡＭコア３４にて保持する要素の範囲とを比較し（ステップ１１０７）、その比較結果によって（ステップ１１０８）、種々の処理を実行する。

【００４０】

まず、削除要求のあった範囲が、自己の保持する要素の範囲よりも後ろである場合（図１１の「Ａ」および図１２（ａ）参照）には、ＭＰＵ３６は何ら処理を実行しない（ステップ１１０９参照）。削除要求のあった範囲が、自己の保持する要素の後方に重なって位置している場合（図１１の「Ｂ」および図１２（ｂ）参照）には、ＭＰＵ３６は、割り付け領域サイズを更新する（ステップ１１１０）。すなわち、削除要求範囲の先頭（矢印１２０１参照）から、自己のＲＡＭコ

ア３４にて保持する要素の範囲の末尾（矢印１２０２参照）までがガーベージとなるように、割り付け領域サイズが変更される。

【００４１】

その一方、削除要求のあった範囲が、自己の保持する要素の範囲よりも前方である場合（図１１の「Ｃ」および図１２（ｃ）参照）には、ＭＰＵ３６は、論理開始アドレスを、削除要求のあったサイズ分だけ減じるように、論理開始アドレスを更新する（ステップ１１１１）。さらに、削除要求のあった範囲が、自己の保持する要素の範囲よりも前方で、かつ、一部だけ重なる場合（図１１の「Ｄ」および図１２（ｄ）参照）には、ＭＰＵ３６は、論理開始アドレスを、削除要求のあった範囲の先頭の値に変更するとともに、物理開始アドレスを、削除要求のあった範囲の末尾の値「＋１」に対応する物理アドレスに変更する（ステップ１１１２）。次いで、ＭＰＵ３６は、割り付け領域サイズを更新する（ステップ１１１３）。

【００４２】

また、削除要求のあった範囲が、自己の保持する要素の範囲を包含する場合（図１１の「Ｅ」および図１２（ｅ）参照）には、ＭＰＵ３６は、当該空間ＩＤに関する種々のデータを、空間ＩＤ管理テーブルから削除する（図１３のステップ１１１４）。最後に、削除要求のあった範囲が、自己の保持する要素の範囲に包含される場合（図１１の「Ｆ」および図１２（ｆ）参照）には、空間ＩＤ管理テーブルを二つに分割して、削除範囲より前方に関する種々のデータと、削除範囲より後方に関する種々のデータに関するものを生成する（ステップ１１１５）。或いは、ＭＰＵ３６は、自己のＲＡＭ３４に関して、ガベージコレクションを時刻しても良い。

【００４３】

このようにして、ＣＰＵ１２からの単一命令（ある空間ＩＤの削除命令）に応答して、各メモリモジュール１４が動作して、所定のメモリモジュールにて必要な処理が並列的に実行される。

次に、ある空間ＩＤを有する配列の末尾に、ある要素を追加する場合につき簡単に説明する。図１４は、ある空間ＩＤの配列の末尾に要素を追加するというイ

ンストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。図１４のステップ１４０１～ステップ１４０６は、図１１のステップ１１０１～ステップ１１０６に対応する。次いで、各メモリモジュール１４のＭＰＵ３６は、追加すべき要素を、自己のＲＡＭコア３４に記憶すべきか否かを判断する（ステップ１４０７）。これは、ＭＰＵ３６が、自己の空間ＩＤ管理テーブルを参照することにより実現できる。ステップ１４０７にてイエス(Yes)と判断された場合には、空間ＩＤ管理テーブル中の必要な値を更新し（たとえば、割り付け領域サイズを、追加する要素数に応じて変更する）、次いで、ＲＡＭセル中の所定の領域に、追加すべき要素を書き込む（ステップ１４０９）。或いは、空間ＩＤ管理テーブルの種々の値を生成して、対応するＲＡＭセル中の領域に、追加すべき要素が書き込まれても良い。

【００４４】

次いで、ＭＰＵ３６は、空間ＩＤ管理テーブル中の当該空間ＩＤに関連する「全サイズ」の値を更新する（ステップ１４１０）。ステップ１４０７においてノー(No)と判断された場合にも、空間ＩＤ管理テーブル中の関連する「全サイズ」の値が更新される。

配列中の任意の位置に要素を追加する場合にも、削除要求と略同等の処理が、各メモリモジュール１４にて実行される。

【００４５】

[多空間メモリのより具体的な動作：配列の結合および分割]

次に、図１５（ａ）に示すように、複数の配列を結合したり、或いは、図１５（ｂ）に示すように、単一の配列を複数の配列に分割する場合につき説明を加える。本実施の形態にかかるコンピュータシステム１０においては、ある空間ＩＤ（図１５（ａ）においては空間ＩＤ「１００」）を有する配列、および／または、他の空間ＩＤ（図１５（ｂ）においては空間ＩＤ「１００」）を有する配列が、単一のメモリモジュールのＲＡＭコアに収容されていても良いし、或いは、複数のメモリモジュールのＲＡＭコアに収容されていても良い。

図１６は、空間ＩＤ「１０」を有する配列および空間ＩＤ「１１」を有する配列、並びに、これらがメモリモジュール中に収容された状態を示す図である。図

１６（ａ）においては、その空間ＩＤが「１０」であり、かつ、各要素のサイズが１０ワードである配列１５０１が示されている。この配列１５０１中の要素は、メモリモジュール１４－１ないし１４－ｘに収容されている。また、図１６（ｂ）においては、その空間ＩＤが「１１」であり、かつ、各要素のサイズが１０ワードである配列１５１０が示されている。この配列１５１０の要素も、メモリモジュール１４－１ないし１４－ｘに収容されている。

【００４６】

ＣＰＵ１２が、制御信号ライン２５を介して、「空間ＩＤ「１０」の配列と空間ＩＤ「１１」の配列とを結合する」旨のインストラクションを発すると、各メモリモジュール１４は、これを受理して、自己の保持しているデータの空間ＩＤに関するインストラクションであるか否かを判断する。これらの処理は、図１１のステップ１１０１ないしステップ１１０６と略同様である。

次いで、自己の保持しているデータの空間ＩＤが、インストラクションに関連している場合には、メモリモジュールのＭＰＵは、以下の手順にしたがって、配列の結合を実現する。

上記図１６に示す場合に、関連する各メモリモジュール１４は、空間ＩＤ「１０」および空間ＩＤ「１１」の双方の要素を保持している場合に、空間ＩＤ「１１」に関する空間ＩＤ管理テーブルの値を更新する。より具体的には、空間ＩＤ「１０」に関する「全サイズ」の値を参照して、その論理開始アドレスを再度算出する（たとえば、図１７の符号１７０１、１７０２参照）。また、関連する各メモリモジュールは、空間ＩＤ管理テーブル中の「全サイズ」の値を、二つの配列をくみ合わせたサイズに対応するものに更新する（たとえば、図１７の符号１７０３参照）。図１７は、このようにして得られた配列１７１０、および、各メモリモジュール１４－１～１４－ｘにおける空間ＩＤ管理テーブル（たとえば、符号１７１１、１７１２参照）を示す図である。

【００４７】

図１８は、空間ＩＤ「１０」を有する配列を、空間ＩＤ「１０」を有する配列と、空間ＩＤ「１１」を有する配列に分割する一例を示す図である。図１８（ａ）に示す、空間ＩＤ「１０」を有する配列の分解点を定め、分解点より前方に位

置する要素を空間ＩＤ「１０」の配列とするとともに、分解点より後方に位置する要素を空間ＩＤ「１１」の配列とする。

【００４８】

この場合にも、ＣＰＵ１２が、制御信号ライン２５を介して、「空間ＩＤ「１０」の配列を、分解点を境にして、空間ＩＤ「１０」の配列と空間ＩＤ「１１」の配列とに分解する」旨のインストラクションを発すると、各メモリモジュール１４は、図１１のステップ１１０１ないしステップ１１０６に略対応する処理を実行し、メモリモジュールのうち、インストラクションに関連するもの（図１８の例では、メモリモジュール１４－１～１４－ｘ）が、所定の処理を実行する。たとえば、ＭＰＵ３６は、分解点より後方に位置する要素を収容している場合に、空間ＩＤ「０１１」に関する空間ＩＤ管理テーブル中の種々の値を作成するとともに、空間ＩＤ「０１０」に関する空間管理ＩＤテーブルのうち、全サイズに関する値を更新する。また、分解点より前方に位置する要素のみを収容している場合にも、メモリモジュールのＭＰＵ３６は、空間ＩＤ「０１０」に関する空間管理ＩＤテーブルのうち、全サイズに関する値を更新する。図１９は、このようにして得られた配列１９０１、１９０２、および、各メモリモジュール１４－１～１４－ｘにおける空間ＩＤ管理テーブル（たとえば、符号１９１１、１９１２および１９１３参照）を示す図である。

【００４９】

[多空間メモリのより具体的な動作：パラレルコピー]

次に、多空間メモリの下で、場合によっては組み替え可能バスを利用したパラレルコピーにつき、簡単に説明を加える。

たとえば、ＣＰＵ１２からの単一のインストラクションにしたがって、図２０に示すように、一方のメモリモジュール群１４０から、他のメモリモジュール群１４１へのデータのパラレルコピーを実現することができる。パラレルコピーには以下の態様が考えられる。

【００５０】

（１）一方のメモリモジュール群１４０には単一のメモリモジュールが含まれ、他方のメモリモジュール群には、複数のメモリモジュールが含まれる場合。

（２）一方のメモリモジュール群１４０に、複数のメモリモジュールが含まれ、他方のメモリモジュール群にも、複数のメモリモジュールが含まれる場合。

【００５１】

前者においては、コピー元の要素を収容しているメモリモジュール１４のＭＰＵ３６は、ＣＰＵ１２から制御信号ライン２５を介して与えられたインストラクション（たとえば、ある空間ＩＤを有する配列中の所定の要素を、空間ＩＤ８、９、１０の配列としてコピーせよという指令）を受理して、ＲＡＭコア３４から指定された要素を所定のバス上に出力する。その一方、コピー先となるＭＰＵ３６も、同一のインストラクションの受理に応答して、バスから出力された要素を受理して、これをＲＡＭコア３４の所定の領域に記憶するとともに、自己の空間ＩＤ管理テーブルを更新する。

【００５２】

後者においては、複数のバスを利用して、一方のメモリモジュール群１４０中のメモリモジュールからのデータを、それぞれ、他方のメモリモジュール群１４１の対応するメモリモジュールに与えることが可能である。この場合には、ＣＰＵ１２は、スイッチ２８およびスイッチ３０を、所定のメモリモジュール間のデータの授受が可能なように制御すれば良い。

【００５３】

［多空間メモリのより具体的な動作：隠れ更新など］

本実施の形態にかかる多空間メモリを用いて、添字変換により、入力された添え字を変換して、変換済みの添え字によって、配列を指定し、さらに、配列の要素に値を修飾することができる。ある処理が終了して、コミットすることにより、添字変換や値修飾が不要となったときに、各メモリモジュールのＭＰＵは、当該配列に関する空間ＩＤ管理テーブルを書きかえてリマッピングを実行することにより、瞬時に添字変換を解消することができる。その一方、値修飾自体は、実際のＲＡＭコアに記憶された要素を更新する必要があるため、時間を要する。したがって、各メモリモジュールにおいて、変換済フラグを設け、値修飾が反映された要素が、実際にＲＡＭコアに記憶した後に、当該要素に対応するフラグを「１」にセットされる。このようにすれば、あるプロセスにおいて、変換フラグを

参照して、これが「１」である場合には、値修飾を経る必要がなく、その一方、変換フラグが「０」であるバイには、値修飾を経る必要があることを容易に知ることができる。したがって、実質的にコミットを瞬時に実現することができる。

さらに、本実施の形態にかかる多空間メモリを用いれば、図２１に示すように、ネスト構造の値修飾についても、変換済フラグを設け、この変換済フラグを参照することにより、値修飾を経る必要の有無を知ることが可能となる。

【００５４】

［多空間メモリおよび組み替え可能バスの利用：ソート（その１）］

本実施の形態においては、多空間メモリおよび組み替え可能バスを利用することにより、ＣＰＵ１２からの単一のインストラクションに基づき、並列的にソート処理を実行することが可能となる。以下、本実施の形態における並列的なソート処理につき説明を加える。

図２３および図２４は、本実施の形態にかかるソート処理の流れを説明するための図である。このソート処理では、大きく分けて、図２３に示す処理（存在数の確定および累計の算出）と、図２４に示す処理（レコード番号の転送）とに分けて考えることができる。

【００５５】

この実施の形態にかかるソート処理を実現するために、本実施の形態においては、レコード番号を格納したレコード番号配列、ある項目に関する実際の項目値を格納した値リスト、および、レコード番号配列からの値（レコード番号）を入力として、対応する値リストの格納位置を示すポインタ値を出力するように構成された値リストへのポインタとを利用している。すなわち、レコード番号から、対応する位置の値リストへのポインタ値が参照され、そのポインタ値にしたがって、実際の項目値が指定されるようになっている（図２５参照）。

まず、ＣＰＵ１２が、必要なインストラクションを、制御信号ライン２５を介して、各メモリモジュール１４に与えると、各メモリモジュールにて、図１１のステップ１１０１ないしステップ１１０６に略同等の処理が実行される。また、関連するメモリモジュールのうち、レコード番号を格納したメモリモジュールからの通知にしたがって、ＣＰＵ１２は、レコード番号を格納した一連のメモリモ

ジュール（第１のメモリモジュール群２３０１）の出力を、あるバス（「第１のバス」と称する）に接続するように、スイッチ２８、３０を制御する。

【００５６】

次いで、値リストへのポインタ配列を格納したメモリモジュールからの通知にしたがって、ＣＰＵ１２は、上記値リストへのポインタ配列を格納した一連のメモリモジュール（第２のメモリモジュール群２３０２）の出力を、あるバス（「第２のバス」と称する）に接続するように、スイッチ２８、３０を制御する。

さらに、他の一連のメモリモジュール（第３のメモリモジュール群２３０３）においては、値リストへのポインタと同一サイズ（同じ要素数）の「存在数配列」のための領域が確保され、かつ、各要素が「０」に初期化される。さらに、第３のメモリモジュール群の入力を、上記第２のバスと接続する。

【００５７】

次いで、レコード番号配列の先頭から順に、レコード番号が第１のバスに送出される。これは、第１のメモリモジュール群２３０１において、各メモリモジュールのＭＰＵ３６が、空間ＩＤ管理テーブルを参照して、自己が第１のバスにデータを出力するタイミングを検出して、所定のレコード番号を送出することにより実現される。

レコード番号は、第１のバスを介して、第２のメモリモジュール群２３０２を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する値リストへのポインタ配列に関連するレコード番号が入力されたことを検出し、当該入力に対応するポインタ値を第２のバスに出力する。

【００５８】

ポインタ値は、第２のバスを介して、第３のメモリーのジュール群を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する値リストのポインタ配列に関連するポインタ値が与えられたことを検出し、存在数配列において、ポインタ値に対応する位置の要素をインクリメントする。この動作を繰り返すことにより、項目値が何度レコード番号により指されているか（ポイントされているか）

を知ることができる。

上記存在数配列のための一連の処理が終了すると、ソートされたレコード番号を格納する配列を作成するために、一連のメモリモジュールに、一定の領域が確保される。この一連のメモリモジュールを、第４のメモリモジュール群２３０４と称する。ＣＰＵ１２は、先の処理に利用した第３のメモリモジュール群の出力と、第４のメモリモジュール群の入力とを、バス（「第３のバス」と称する）を介して接続するように、スイッチ２８、３０を制御する。

【００５９】

このような準備が終了した後に、ソート処理が実行される。より具体的には、レコード番号配列の先頭から、レコード番号が第１のバスを介して、第２のメモリモジュール群を構成するメモリモジュールに与えられる。第２のメモリモジュール群中の所定のメモリモジュールにおいては、ＭＰＵ３６がレコード番号の受理に応答して、ポインタ値を、第２のバスを介して、第３のモジュール群に伝達する。

次いで、第３のメモリモジュール群のうち、所定のメモリモジュールにおいて、ＭＰＵ３６が、ポインタ値に基づき、関連する存在数配列を参照して、レコード番号の格納位置を決定する。これにより、レコード番号およびその格納位置が、が当該メモリモジュールから、第３のバスに送出される。したがって、第４のメモリモジュール群の所定のメモリモジュールにおいて、ＭＰＵ３６が、レコード番号を、所定の格納位置に配置する。この処理を繰り返すことにより、第４のメモリモジュール群に、ソートされたレコード番号の配列（図２４の符号２４１０）を作成することができる。

【００６０】

たとえば、図２３に示す処理を、パイプライン処理にすることができる。すなわち、第１のバスにおいて、あるレコード番号「ｐ」が伝達されている際に、第２のバスにおいては、レコード番号「ｐ－１」に関するポインタ値「Ｐ（ｐ－１）」が伝達され得る。また、同様に、図２４に示す処理も、パイプライン処理にすることが可能である。この場合にも、第１のバスにおいて、あるレコード番号「ｐ」が伝達されている際に、第２のバスにおいては、レコード番号「ｐ－１」

に関するポインタ値「Ｐ（ｐ－１）」が伝達され得る。さらに、同じタイミングで、第３のバスにおいては、レコード番号「ｐ－１」に関する格納位置が伝達され得る。

【００６１】

このようなパイプライン処理の処理時間につき、以下のような結果が得られた。まず、図２３の処理に関して、第１のバスないし第４のバスが、それぞれ、１２８ビットであり、それぞれ、１２．８ＧＢ／秒の転送能力があると考え、また、レコード番号やポインタ値が、それぞれ、３２ビット整数であると仮定した。いま、レコード数が１０億個の場合に、上記処理では、４０億バイトの転送が発生するが、パイプライン処理を実行するため、４Ｇ／１２．８Ｇ＝０．３１２５秒にて完了することが分かった。

同様に、図２４の処理に関して、同様の転送能力およびデータサイズを仮定すると、レコード数が１０億個の場合に、８０億バイトの転送が発生するが、本実施の形態によれば、パイプライン処理の実行により、８Ｇ／１２．８Ｇ＝０．６２５秒にて処理を完了することができる。

【００６２】

［多空間メモリおよび組み替え可能バスの利用：ソート（その２）］

次に、他の手法によるソート処理につき簡単に説明を加える。このソート処理においても、まず、レコード番号配列を格納したメモリモジュールからなる第１のメモリモジュール群（図２６の符号２６０１参照）の出力と、第１のバスとが接続され、かつ、値リストへのポインタ配列を格納したメモリモジュールからなる第２のメモリモジュール群２６０２の入力が、第１のバスと接続される。これにより、第１のメモリモジュール群２６０１の出力が、第１のバスを介して、第２のメモリモジュール群２６０２に伝達可能となる。

その一方、第２のメモリーモジュール群２６０２と、同一の数の空間ＩＤを有する配列の領域が、第３のメモリモジュール群２６０３に確保されるとともに、第２のメモリモジュール群２６０２の出力と、第３のメモリモジュール群の入力とが、第２のバスを介して接続される。

【００６３】

次いで、第１のメモリモジュール群２６０１において、あるレコード番号を収容するメモリモジュールのＭＰＵ３６が、当該レコード番号を、第１のバスに送出すると、第２のメモリモジュー群２６０２の所定のメモリモジュールにおいて、ＭＰＵ３６がこの受理に応答して、対応するポインタ値から、空間ＩＤを算出し、レコード番号および空間ＩＤを、第２のバスに送出する。

【００６４】

第３のメモリモジュール群において、当該空間ＩＤおよびレコード番号に基づき、所定のメモリモジュール３６が起動し、当該空間ＩＤを有する配列の末尾に、与えられたレコード番号を配置する。このような処理を全てのレコード番号について実行した後に、第３のメモリモジュール群において、各メモリモジュールのＭＰＵ３６は、自己の有する配列を結合するための処理を実行する。このような手法によっても、高速なソート処理を実現することができる。

【００６５】

［多空間メモリおよび組み替え可能バスの利用：検索（その１）］

また、本実施の形態においては、多空間メモリおよび組み替え可能バスを利用することにより、ＣＰＵ１２からの単一のインストラクションに基づき、並列的に検索処理を実行することができる。

図２７および図２８は、本実施の形態にかかる検索処理の流れを説明するための図である。この検索処理のために、レコード番号配列、値リストへのポインタ配列、値リストおよび後述する可否フラグ配列などが利用される。したがって、この例でも、図２５のように、レコード番号、ポインタ値、項目値の順に、値が参照されるようになっている。

【００６６】

まず、ＣＰＵ１２が、必要なインストラクションを、制御信号ライン２５を介して、各メモリモジュール１４に与えると、各メモリモジュールにて、図１１のステップ１１０１ないしステップ１１０６に略同等の処理が実行される。また、関連するメモリモジュールのうち、値リストを格納したメモリモジュールからの通知にしたがって、ＣＰＵ１２は、値リストを格納した一連のメモリモジュール（第１のメモリモジュール群２７０１）の出力を、あるバス（「第１のバス」と

称する）に接続するように、スイッチ２８、３０を制御する。さらに、そのよう素数が値リストのものと同じである可否フラグ配列のための領域が、一連のメモリモジュール（第２のメモリモジュール群２７０２）に確保され、当該第２のメモリモジュール２７０２に属する各メモリモジュールのＭＰＵ３６が、当該領域の要素を「０」に初期化する。

【００６７】

次いで、第２のメモリモジュール群２７０２の入力が、第１のバスに接続される。次いで、ＣＰＵ１２から与えられた検索条件にしたがって、第２のメモリモジュール群の各メモリモジュールにおいて、ＭＰＵ３６が、値リスト中の検索条件に合致する項目値の位置を参照して、可否フラグ配列の対応する値を「１」にセットする。たとえば、検索条件が範囲であれば、二分割法などを用いれば良い。また、その他の条件であれば、要素ごとにその可否を判断すれば良い。

このような処理が終了した後に、検索が実行される。まず、レコード番号配列を格納した一連のメモリモジュール（第３のメモリモジュール群２７０３）の出力を、第１のバスに接続するとともに、値リストへのポインタ配列を格納した一連のメモリモジュール（第４のメモリモジュール群２７０４）の入力を、第１のバスに接続するよう、ＣＰＵ１２は、スイッチ２８、３０を制御する。また、第４のメモリモジュール群２７０４の出力を、第２のメモリモジュール群２７０２の入力とを、第２のバスと接続するように、ＣＰＵ１２は、スイッチ２８、３０を制御する。

【００６８】

さらに、レコード番号の要素数と同じ要素数を有する配列のための領域が、一連のメモリモジュール（第５のメモリモジュール２７０５）に確保され、ＣＰＵ１２は、その入力と、第２のメモリモジュール群２７０２の出力とが、第３のバスを介して接続されるように、スイッチ２８、３０を制御する。

このような処理の後に、レコード番号配列の先頭から順に、レコード番号が第１のバスに送出される。これは、第３のメモリモジュール群２７０３において、各メモリモジュールのＭＰＵ３６が、空間ＩＤ管理テーブルを参照して、自己が第１のバスにデータを出力するタイミングを検出して、所定のレコード番号を送

出することにより実現される。

【００６９】

レコード番号は、第１のバスを介して、第４のメモリモジュール群２７０４を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する値リストへのポインタ配列に関連するレコード番号が入力されたことを検出し、受理したレコード番号および当該入力に対応するポインタ値を第２のバスに出力する。

ポインタ値は、レコード番号とともに、第２のバスを介して、第３のメモリーのジュール群を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する可否フラグ配列の位置と同じ位置を示すポインタ値が与えられたことを検出し、当該ポインタ値が示す可否フラグが、「０」であるか「１」であるかを判断する。次いで、可否フラグが「１」の場合には、関連するレコード番号が、第３のバスを介して、第５のメモリモジュール群２７０５に与えられる。

【００７０】

第５のメモリモジュール群２７０５においては、各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理するヒット情報格納用配列の位置と同じ位置を示すレコード番号が与えられたことを検出し、その位置の要素を「１」にする。このような処理を所定のレコード番号に関して繰り返し、ヒット情報格納用配列にて「１」である要素を取り出すことにより、検索が完了する。

【００７１】

ソート処理と同様に、上記検索処理でも、図２７を参照して説明した処理、および、図２８を参照して説明した処理を、それぞれ、パイプライン処理にて実現することができる。検索処理におけるパイプライン処理の処理時間につき、以下のような結果が得られた。

バスの転送能力、および、各要素のビット数は、ソート処理と同様であると考えた。レコード数が１０億個の場合に、上記検索処理では、８０億バイトの転送が発生するが、パイプライン処理を実行するため、８Ｇ／１２．８Ｇ＝０．６２

４秒にて完了することが分かった。

【００７２】

さらに、本検索処理を用いれば、ＡＮＤ、ＯＲ或いはＮＯＴなどを組み合わせた複数項目の検索を実現することもできる。より具体的には、各項目につき、ヒット情報格納用配列を作成し、これら配列の要素間での論理演算を行えば良い。

たとえば、二つの項目のＡＮＤ或いはＯＲ検索では、ヒット情報格納用配列の要素の転送（１０億バイト）が行われる。したがって、その処理時間は、（１０Ｇ／８）／１２．８Ｇ＝０．０９８秒だけ必要であることが理解できる。

【００７３】

なお、さらに高速化を図るために、ＡＮＤ検索の場合には、二つの検索処理を実行するメモリモジュール群を縦列に接続すれば良い。また、第４のメモリモジュール群と第２のメモリモジュール群を、同一の複数のメモリモジュールにて構成できるように、配列を配置すれば、ボトルネックを解消することができ、これにより、略２倍の処理速度を得ることが可能となる。

【００７４】

本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。

たとえば、前記実施の形態においては、本発明を、コンピュータシステムに適用しているがこれに限定されるものではなく、パーソナルコンピュータなどに接続可能なコンピュータボードに適用することもできる。この場合には、図１において、ＣＰＵ１２、メモリユニット１４、バス２４等がボード上に搭載され、これが、本発明における情報処理ユニットを構成する。

【００７５】

また、ＣＰＵ１２とメモリモジュール１４との間、および／または、メモリモジュール１４間を接続するバスの組の数は、前記実施の形態に限定されるものではなく、コンピュータシステムを搭載する回路基板の大きさ、各バスのビット数などを考慮して適宜決定することができる。また、前記実施の形態においては、メモリモジュールの入出力とバスとの接続を規定するためのスイッチ２８と、Ｃ

ＰＵとメモリモジュールとの間、メモリモジュール間で、バスの切断することができるスイッチ３０とを設けている。スイッチ３０を設けることにより、たとえば、あるバス（図１のバス２４－４参照）を、ＣＰＵモジュール１２とメモリモジュール１４－１とのデータ授受のために利用するとともに、同時に、メモリモジュール１４－２とメモリモジュール１４－３との間のデータ授受のために利用することができる（この場合に、スイッチ３０－５をオフにすれば良い）。したがって、より有効にバスを利用することが可能となっている。しかしながら、バスの組を数を十分に大きくできる場合、或いは、メモリモジュールの数が比較的少ない場合には、スイッチ３０を必ずしも設けなくて良い。

【００７６】

また、本明細書において、制御信号ライン２５を介して、ＣＰＵ１２からのインストラクションが与えられることを記載したが、制御信号ライン２５を介して、インストラクションのほか、クロックなど、各メモリモジュールが同期して作動するための種々の制御信号が与えられ、かつ、各メモリモジュールからＣＰＵ１２への所定の信号（たとえば、エラー信号や、データ受理を示す信号）が与えられていることは言うまでもない。

さらに、本明細書において、一つの手段の機能が、二つ以上の物理的手段により実現されても、若しくは、二つ以上の手段の機能が、一つの物理的手段により実現されてもよい。

【００７７】

【発明の効果】

本発明によれば、分散メモリー型において、単一命令により種々のメモリーに記憶された配列中の要素を入出力し、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供することが可能となる。

【図面の簡単な説明】

【図１】　図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。

【図２】　図２は、本実施の形態にかかるメモリモジュールの概略を示すブロックダイヤグラムである。

【図３】　図３は、単一メモリ空間における一連のデータの配置を示す図である。

【図４】　図４は、本発明に係る多空間メモリにおける一連のデータの配置を示す図である。

【図５】　図５は、本実施の形態におけるアドレスリマッピングを説明するための図である。

【図６】　図６は、本実施の形態における値修飾を説明するための図である。

【図７】　図７は、本実施の形態にかかるメモリモジュール間のパイプライン処理の概略を示す図である。

【図８】　図８は、本実施の形態にかかる多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図９】　図９は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図１０】　図１０は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図１１】　図１１は、ある空間ＩＤ中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図１２】　図１２は、削除される要素と、メモリモジュールにて保持している要素の配置との関係を示す図である。

【図１３】　図１３は、ある空間ＩＤ中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図１４】　図１４は、ある空間ＩＤの配列の末尾に要素を追加するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図１５】　図１５は、本実施の形態にかかる配列の結合および配列の分割を説明するための図である。

【図１６】　図１６は、本実施の形態において、空間ＩＤ「１０」を有する配

列および空間ＩＤ「１１」を有する配列、並びに、これらがメモリモジュール中に収容された状態を示す図である

【図１７】　図１７は、本実施の形態において、配列の結合により得られた配列、および、各メモリモジュールにおける空間ＩＤ管理テーブルを示す図である。

【図１８】　図１８は、本実施の形態において、空間ＩＤ「１０」を有する配列を、空間ＩＤ「１０」を有する配列と、空間ＩＤ「１１」を有する配列に分割する一例を示す図である。

【図１９】　図１９は、本実施の形態において、配列の分割により得られた配列、および、各メモリモジュールにおける空間ＩＤ管理テーブルを示す図である。

【図２０】　図２０は、本実施の形態かかる、一方のメモリモジュール群から、他のメモリモジュール群へのデータのパラレルコピーを示す図である。

【図２１】　図２１は、本実施の形態にかかる変換済みフラグの利用を説明するための図である。

【図２２】　図２２は、本実施の形態にかかる変換済みフラグの利用を説明するための図である。

【図２３】　図２３は、本実施の形態にかかるソート処理の流れを説明するための図である。

【図２４】　図２４は、本実施の形態にかかるソート処理の流れを説明するための図である。

【図２５】　図２５は、本実施の形態において、レコード番号から項目値が特定されるまでのデータの参照手順を示す図である。

【図２６】　図２６は、本実施の形態にかかる他のソート処理の流れを説明するための図である。

【図２７】　図２７は、本実施の形態にかかる検索処理の流れを説明するための図である。

【図２８】　図２８は、本実施の形態にかかる検索処理の流れを説明するための図である。

【符号の説明】

１０	コンピュータシステム
１２	ＣＰＵモジュール
１４	メモリモジュール
１６	固定記憶装置
１８	入力装置
２０	表示装置
２２	レガシーメモリ
２４	バス
２５	制御信号ライン
２６	バス
２８、３０	スイッチ
３２	クロックバッファ
３４	ＲＡＭコア
３６	ＭＰＵ
３８	Ｉ／Ｏ

【書類名】　図面

【図１】

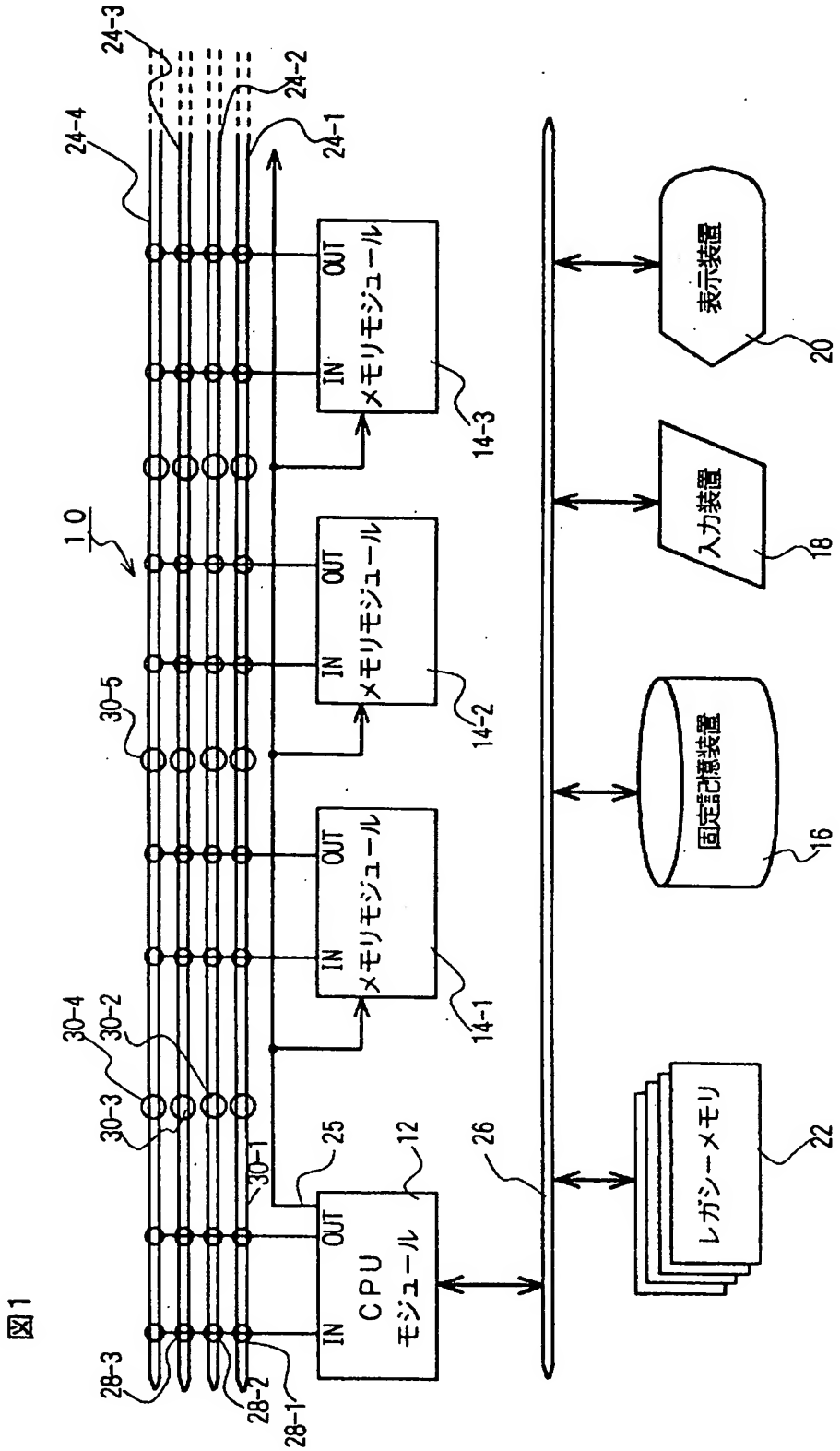

24-3
24-2
24-4
24-1
10
IN
OUT
メモリモジュール
14-3
14-2
14-1
30-5
30-4
30-2
30-3
30-1
25
12
CPU
モジュール
26
28-3
28-2
28-1
表示装置
20
入力装置
18
固定記憶装置
16
レガシーメモリ
22

図1

【図２】

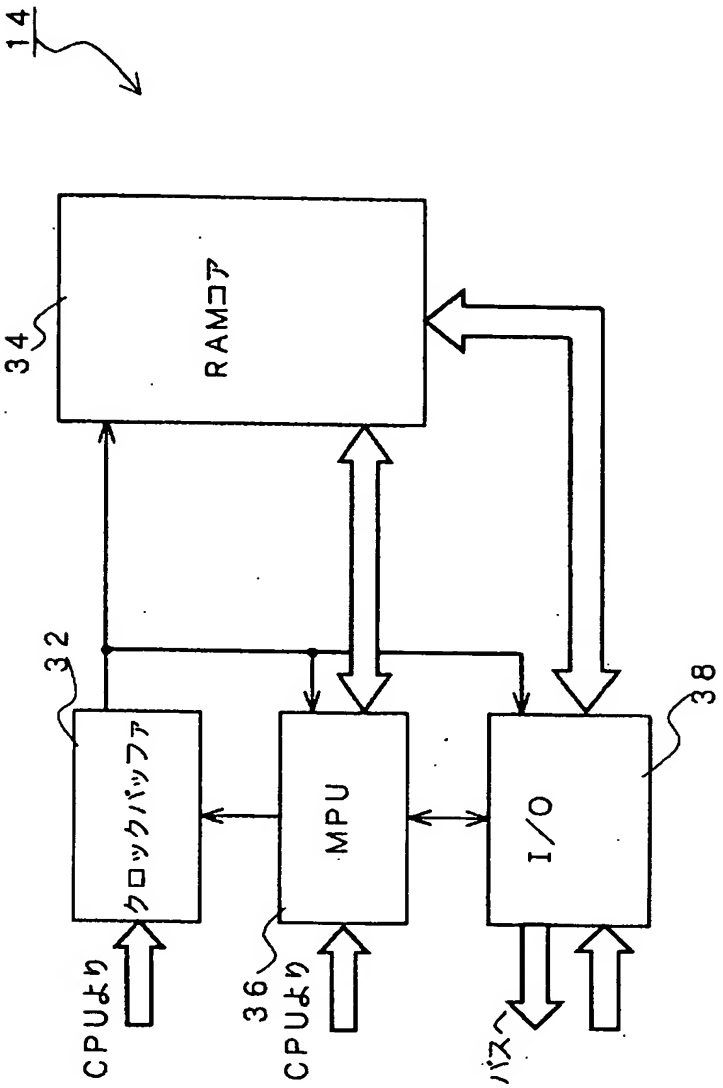


図2

【図３】

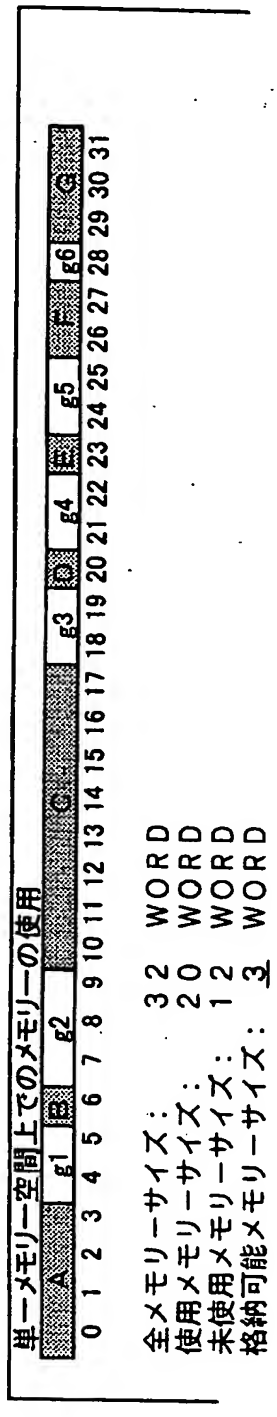

単一メモリー空間上でのメモリーの使用
A
g1
B
g2
C
g3
D
g4
E
g5
F
g6
G
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
全メモリーサイズ：　32 WORD
使用メモリーサイズ：　20 WORD
未使用メモリーサイズ：　12 WORD
格納可能メモリーサイズ：　3 WORD

図3

【図４】

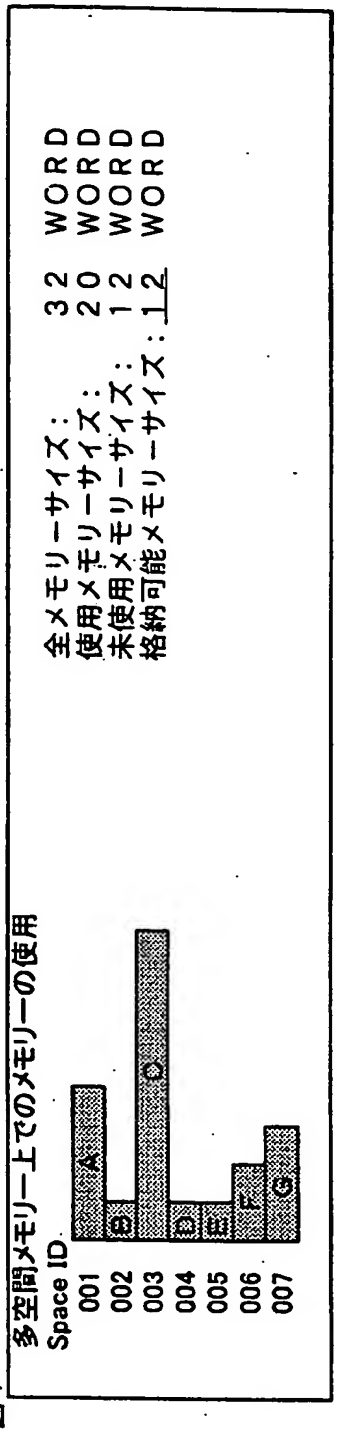

多空間メモリー上でのメモリーの使用
Space ID
001
002
003
004
005
006
007
A
B
C
D
E
F
G
全メモリーサイズ：　32　WORD
使用メモリーサイズ：　20　WORD
未使用メモリーサイズ：　12　WORD
格納可能メモリーサイズ：12　WORD

図4

【図５】

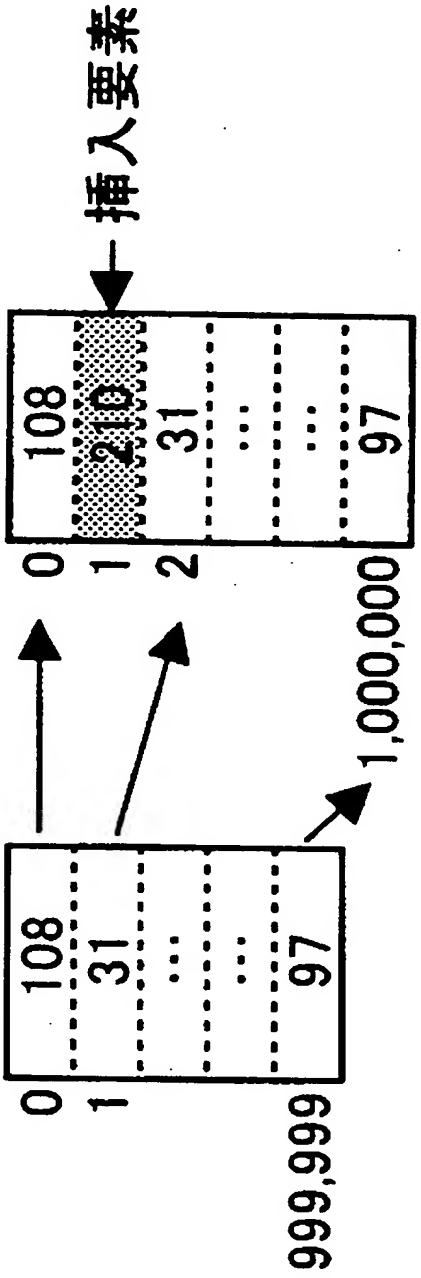


図5

【図６】

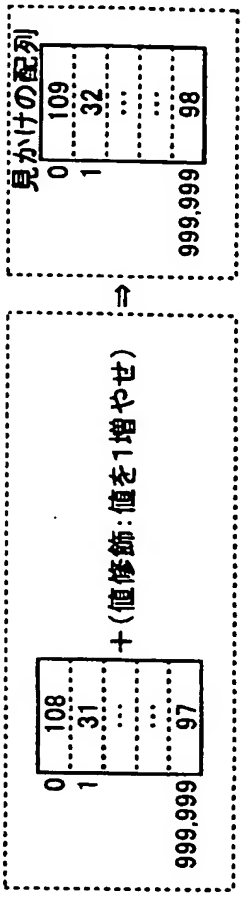


図6

【図７】

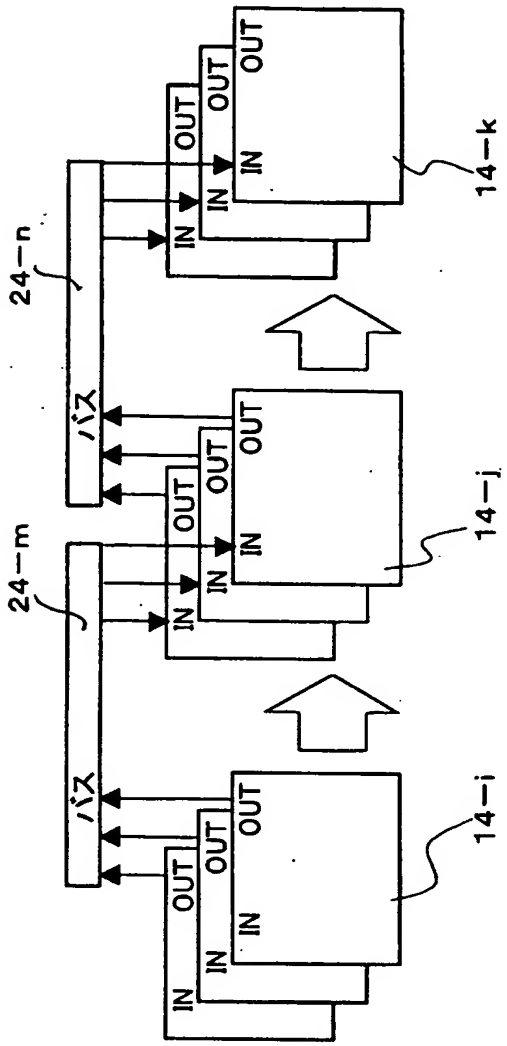


図7

【図８】


(a)
MPU
36
空間ID管理テーブル
物理メモリーセル
34
メモリモジュール
14
(b)
空間ID
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
アクセス制限フラグ
010 100 60 0 164 R
010 160 4 80 164 R
016 0 20 60 20 RW
212 0 100 360 100 RW
アクセス制限フラグについて
R:読み出しのみ可
W:書きこみのみ可
RW:読み書き可能
(c)
物理メモリーセルの内部
物理アドレス
ID
論理アドレス
空間ID 010
空間ID 016
空間ID 010
空間ID 212
0 10 010:0100
1 21 010:0101
59 33 010:0159
60 2323 016:0000
79 3241 016:0019
80 15 010:0159
83 8 010:0163
360 -589 212:0000
459 -1022 212:0099

図8

【図９】

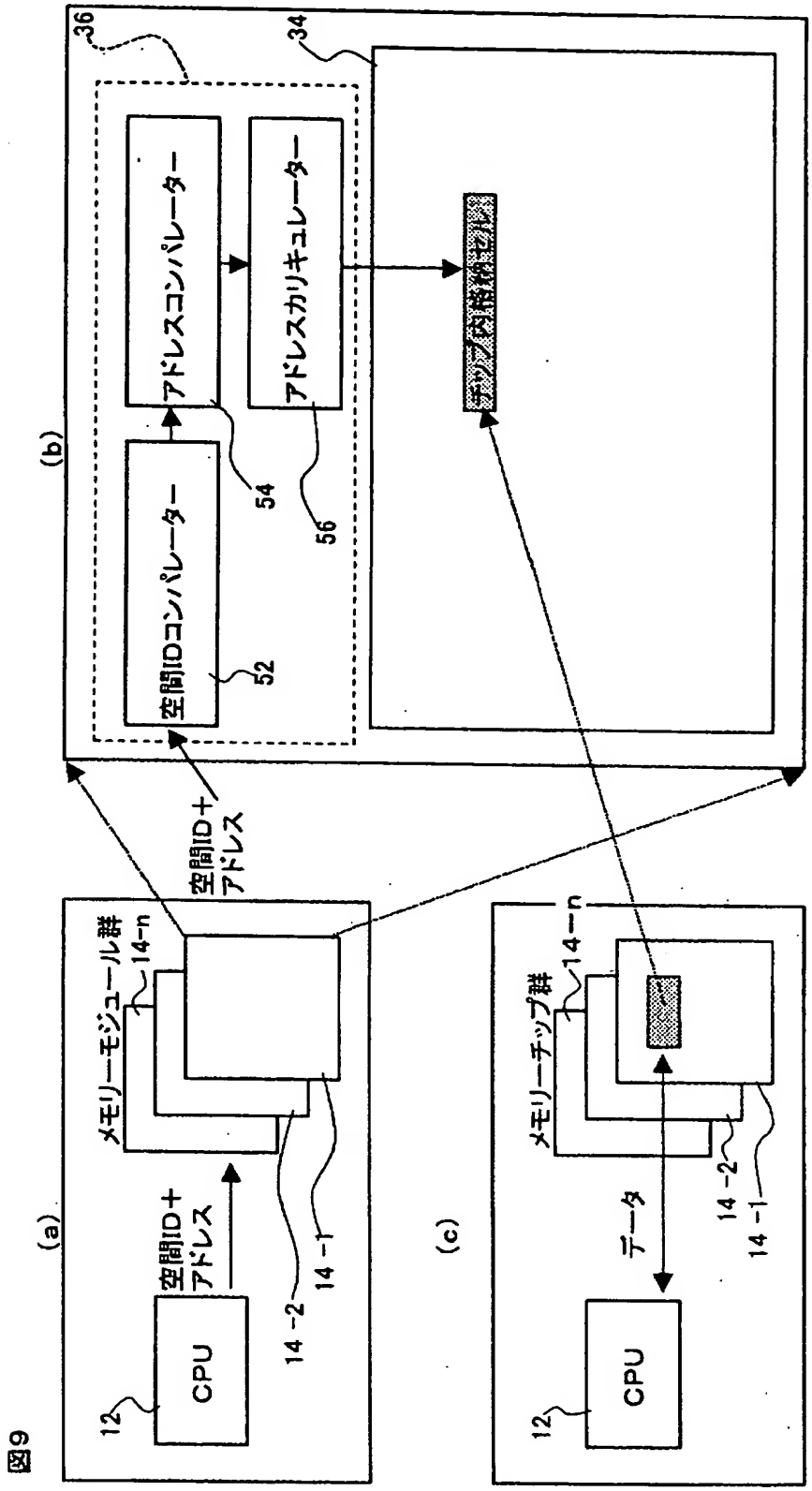

(a)
(b)
(c)
CPU
12
空間ID＋アドレス
メモリーモジュール群
14-n
14-2
14-1
空間IDコンパレーター
52
アドレスコンパレーター
54
アドレスカリキュレーター
56
チップ内格納位置
36
34
メモリーチップ群
データ

図9

【図１０】

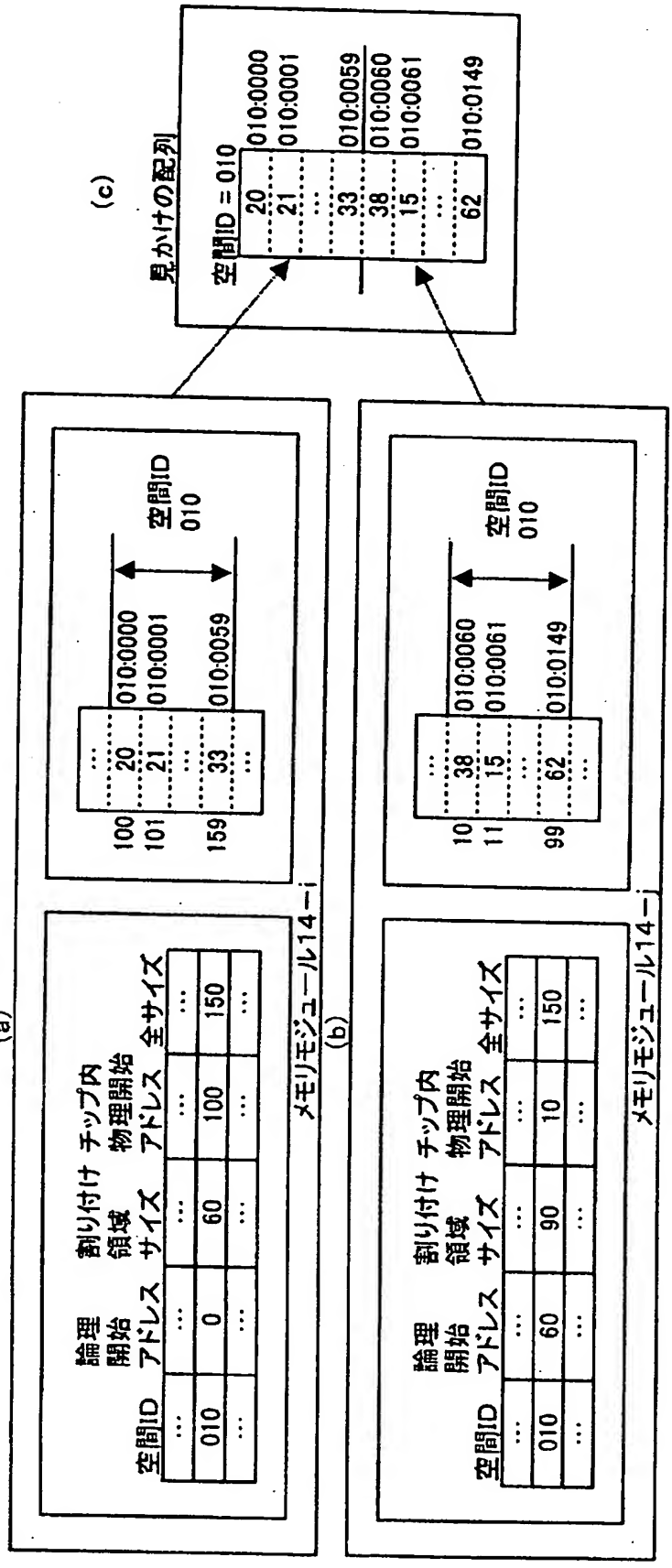
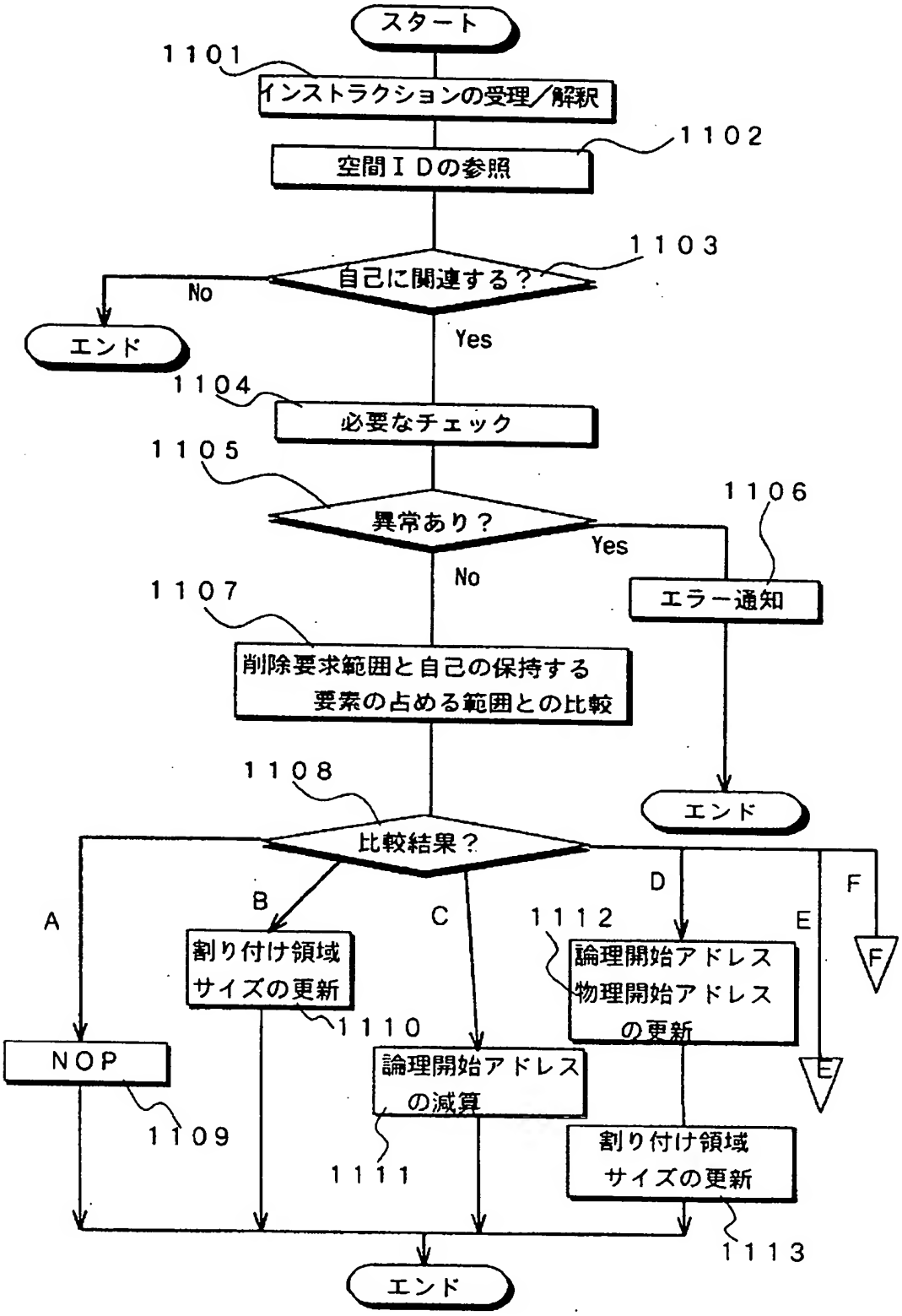

(a)
(b)
(c)
メモリモジュール14－i
メモリモジュール14－j
空間ID
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
010
0
60
100
150
60
90
10
150
見かけの配列
空間ID = 010
20
21
33
38
15
62
010:0000
010:0001
010:0059
010:0060
010:0061
010:0149
100
101
159
10
11
99

図10

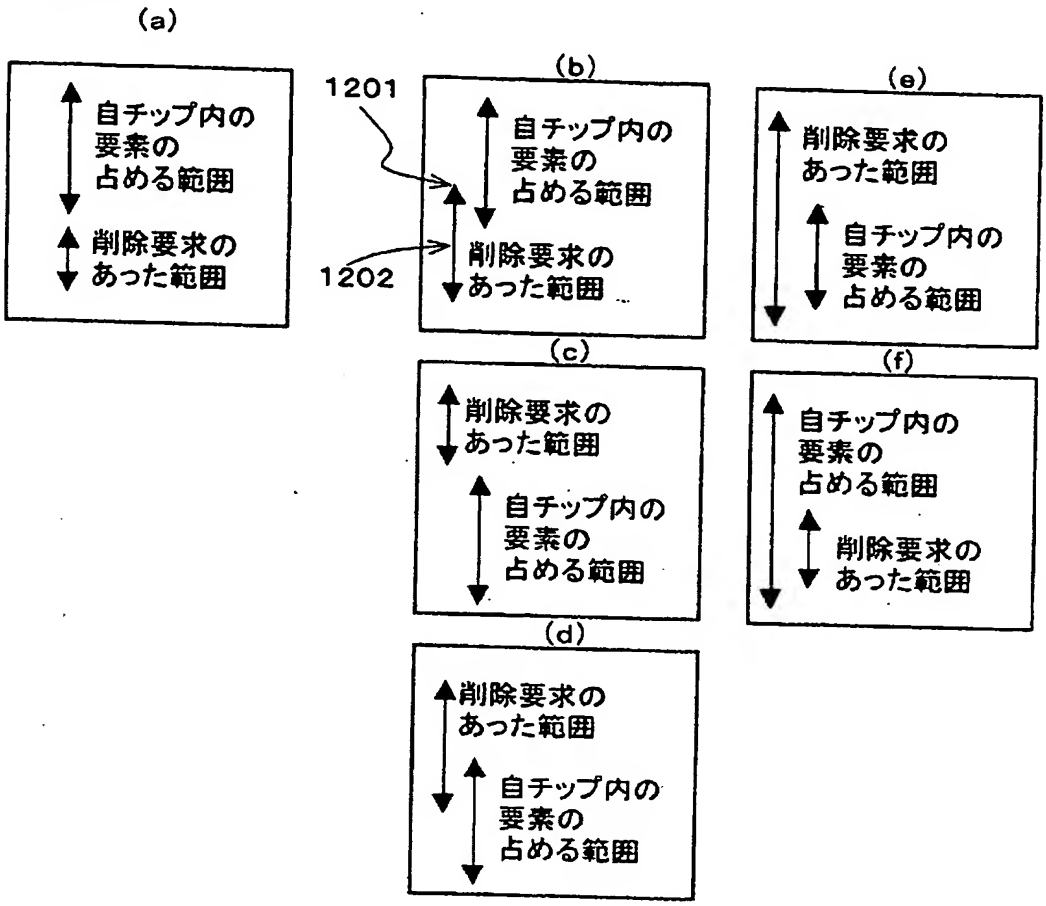
【図１１】

図１１


スタート
1101
インストラクションの受理／解釈
1102
空間ＩＤの参照
1103
自己に関連する？
No
エンド
Yes
1104
必要なチェック
1105
異常あり？
1106
Yes
No
エラー通知
1107
削除要求範囲と自己の保持する
要素の占める範囲との比較
1108
エンド
比較結果？
A
B
C
D
E
F
割り付け領域
サイズの更新
1110
1112
論理開始アドレス
物理開始アドレス
の更新
NOP
1109
論理開始アドレス
の減算
1111
割り付け領域
サイズの更新
1113
エンド

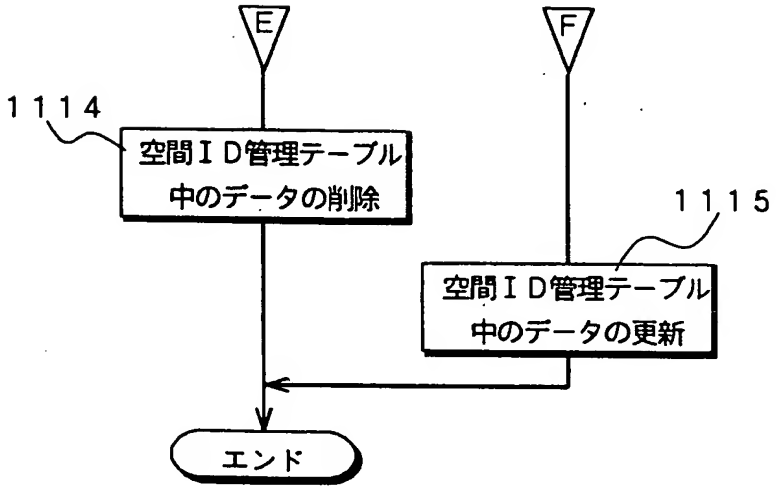
【図１２】

図12



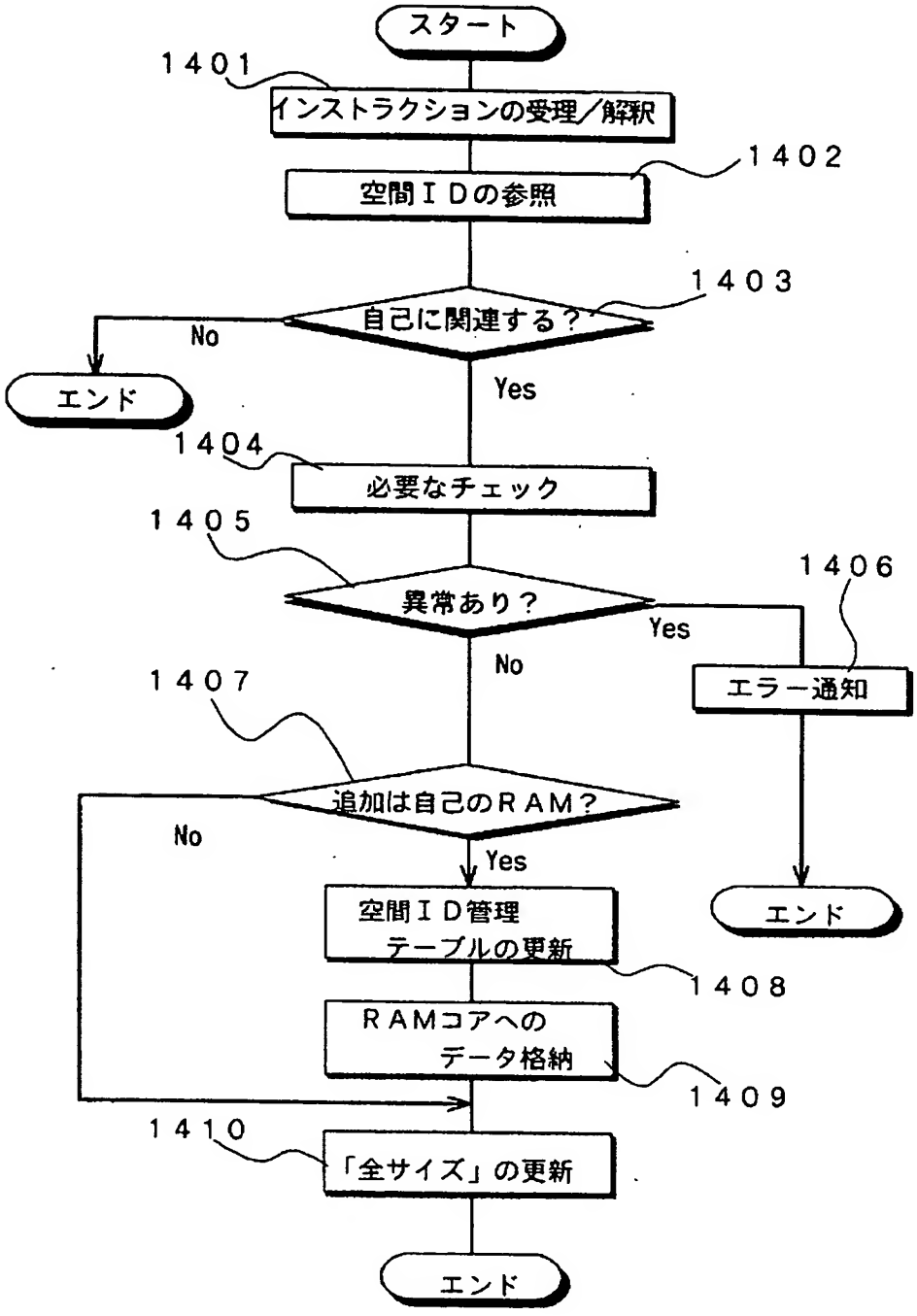
【図１３】

図１３



【図１４】

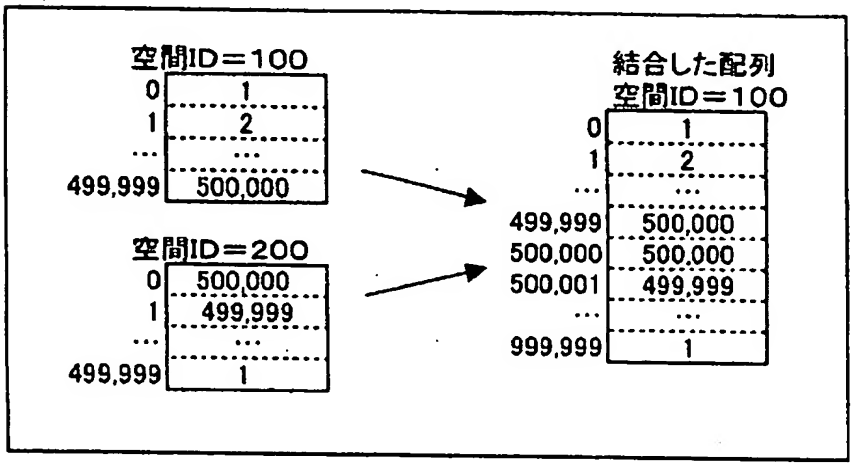
図１４


スタート
1401
インストラクションの受理／解釈
1402
空間ＩＤの参照
1403
自己に関連する？
No
エンド
Yes
1404
必要なチェック
1405
異常あり？
1406
Yes
エラー通知
No
1407
追加は自己のＲＡＭ？
No
Yes
空間ＩＤ管理
テーブルの更新
エンド
1408
ＲＡＭコアへの
データ格納
1409
1410
「全サイズ」の更新
エンド

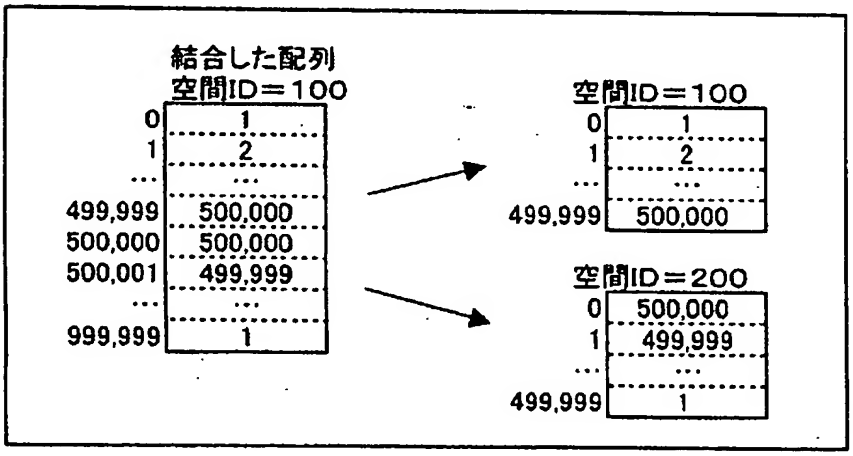
【図１５】

図15

(a)


空間ID＝100
0
1
...
499,999
1
2
...
500,000
空間ID＝200
0
1
...
499,999
500,000
499,999
...
1
結合した配列
空間ID＝100
0
1
...
499,999
500,000
500,000
500,001
...
999,999
1
2
...
500,000
499,999
...
1

(b)


結合した配列
空間ID＝100
0
1
...
499,999
500,000
500,000
500,001
...
999,999
1
2
...
500,000
499,999
...
1
空間ID＝100
0
1
...
499,999
1
2
...
500,000
空間ID＝200
0
1
...
499,999
500,000
499,999
...
1

【図１６】

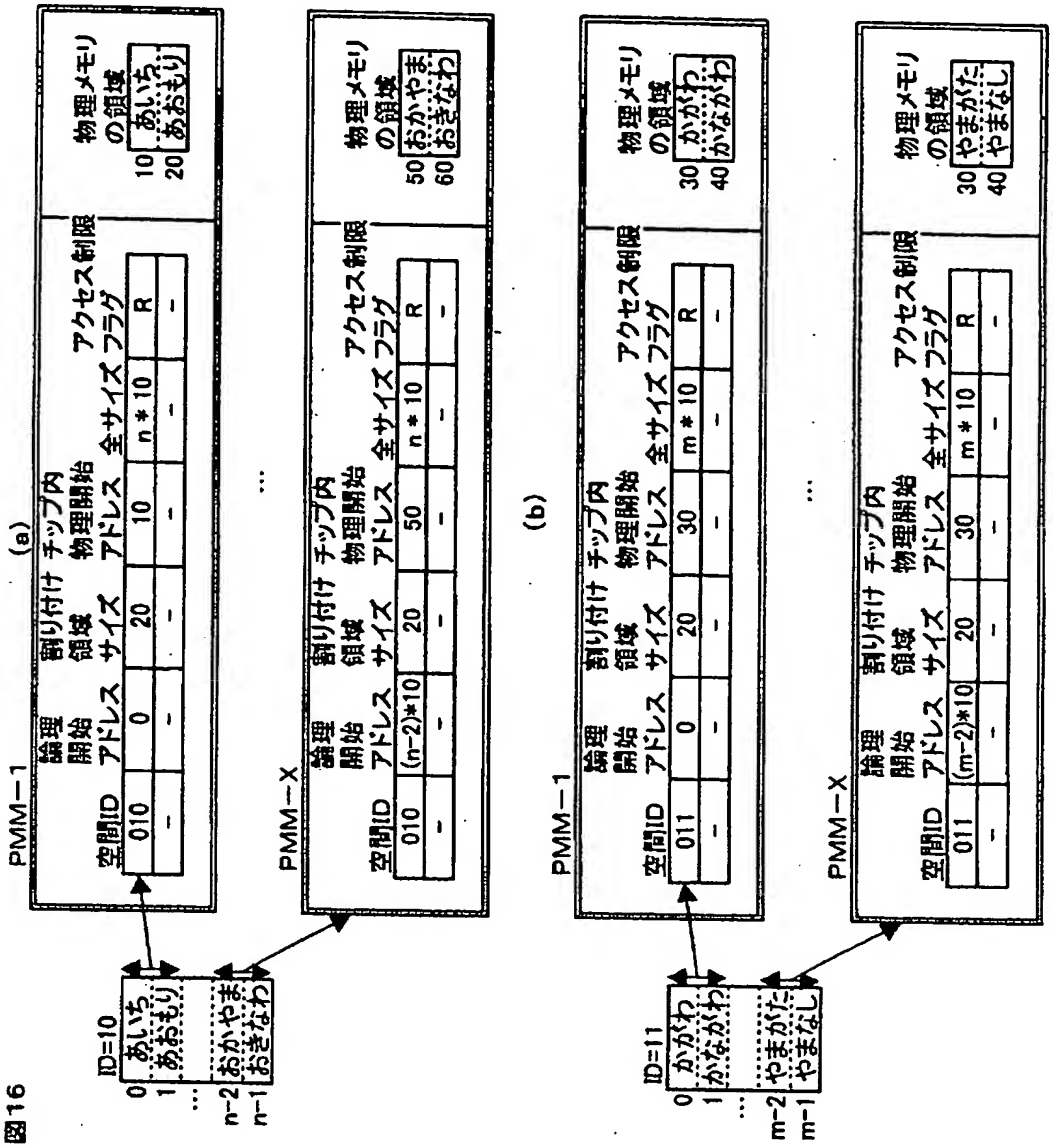
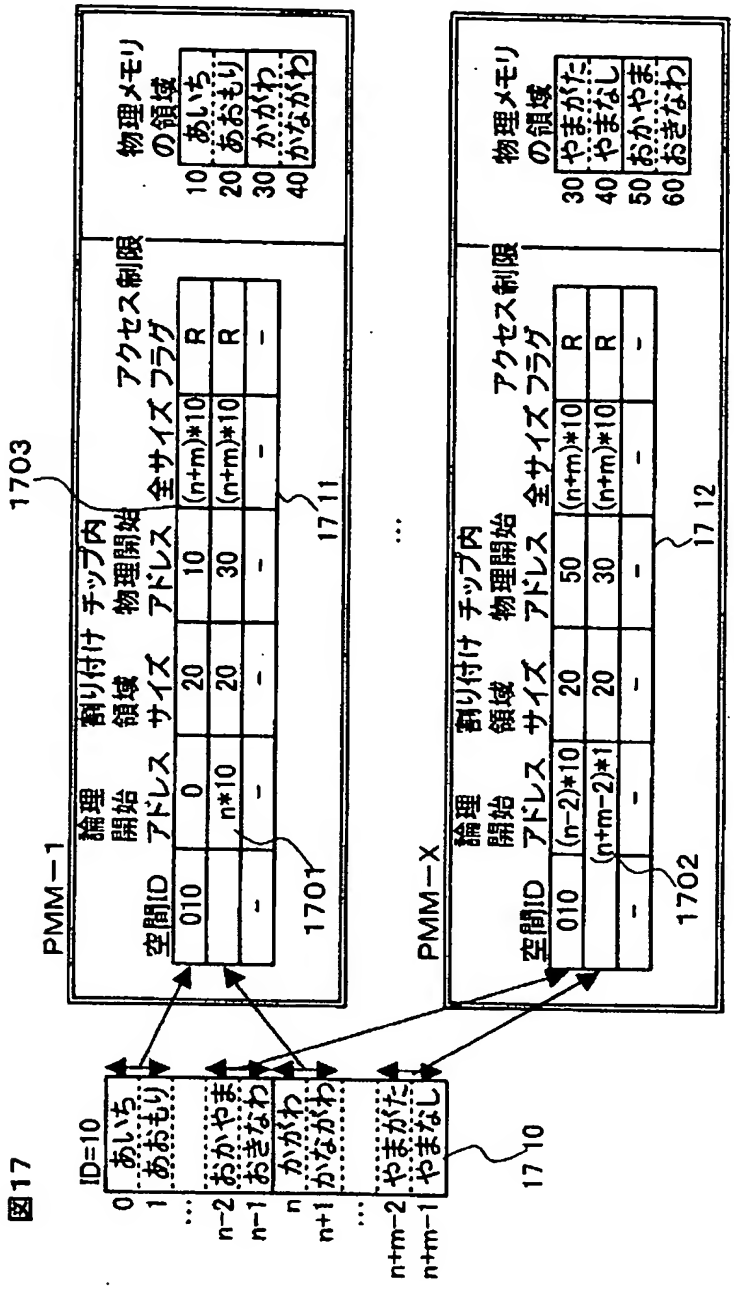

(a)
PMM−1
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
アクセス制限フラグ
物理メモリの領域
空間ID
010
0
20
10
n＊10
R
10
あいち
20
あおもり
PMM−X
(n−2)＊10
50
60
おかやま
おきなわ
ID=10
0
1
n−2
n−1
(b)
PMM−1
011
30
40
m＊10
かがわ
かながわ
PMM−X
(m−2)＊10
やまがた
やまなし
ID=11
0
1
m−2
m−1

図16

【図１７】


1703
PMM−1
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
アクセス制限フラグ
空間ID
010
0
20
10
(n+m)*10
R
n*10
20
30
(n+m)*10
R
−
−
−
−
−
1701
1711
物理メモリの領域
10 あいち
20 あおもり
30 かがわ
40 かながわ
PMM−X
(n−2)*10
(n+m−2)*1
50
30
1702
1712
30 やまがた
40 やまなし
50 おかやま
60 おきなわ
ID=10
0 あいち
1 あおもり
n−2 おかやま
n−1 おきなわ
n かがわ
n+1 かながわ
n+m−2 やまがた
n+m−1 やまなし
1710
図17

【図１８】

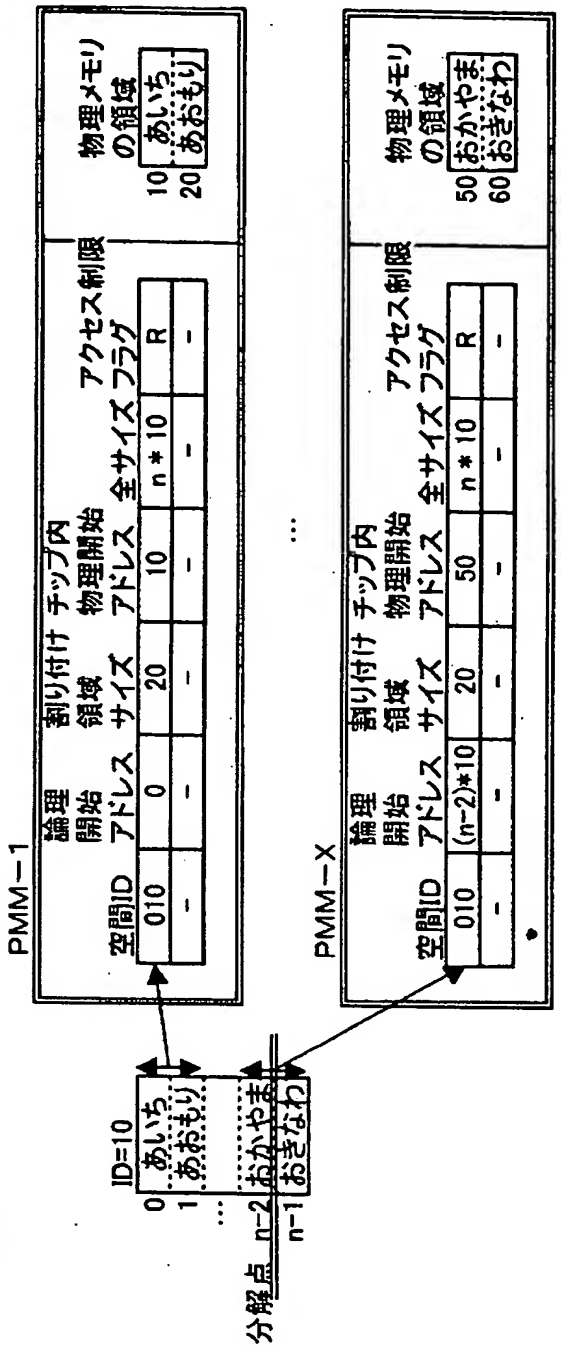

PMM−1
空間ID
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
アクセス制限フラグ
物理メモリの領域
010
0
20
10
n＊10
R
10 あいち
20 あおもり
PMM−X
(n−2)*10
50
60 おかやま
おきなわ
ID=10
0 あいち
1 あおもり
n−2 おかやま
n−1 おきなわ
分解点

図18

【図１９】

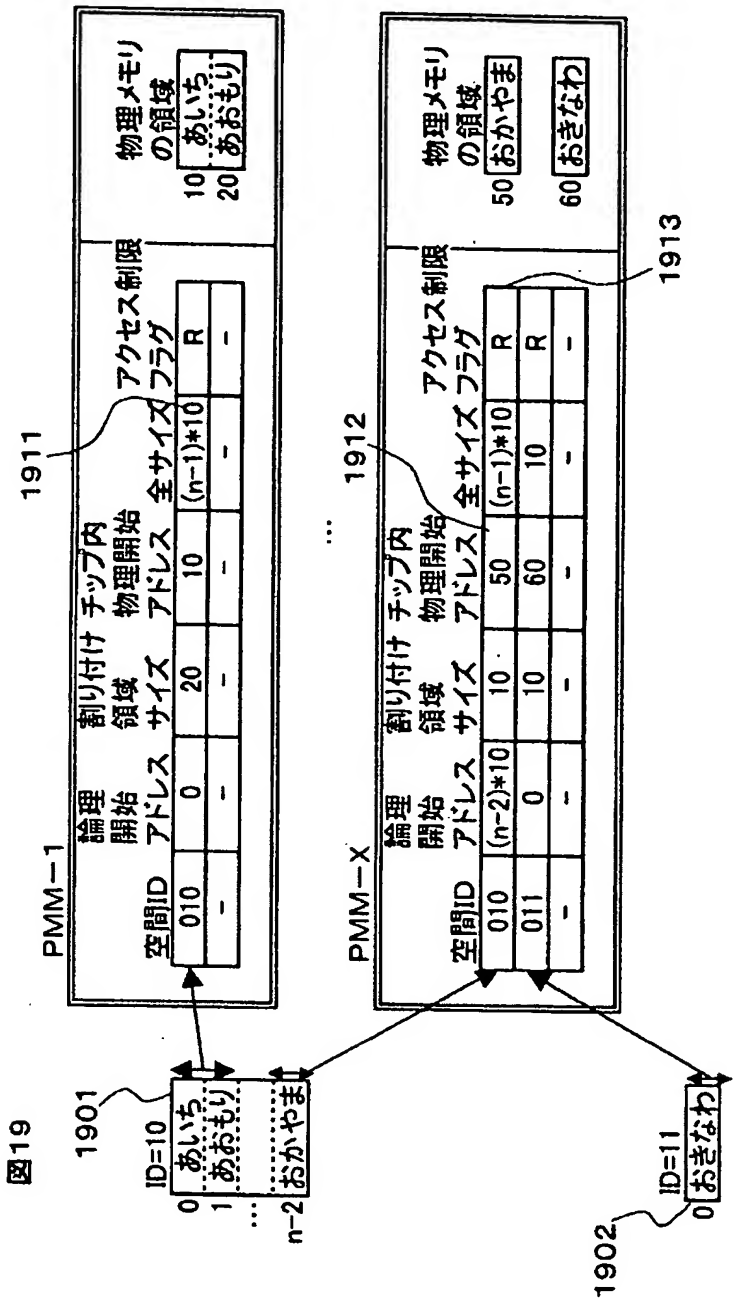

PMM−1
空間ID
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
アクセス制限フラグ
物理メモリの領域
010
0
20
10
(n−1)*10
R
10 あいち
20 あおもり
PMM−X
010
(n−2)*10
10
50
(n−1)*10
R
50 おかやま
011
0
10
60
10
R
60 おきなわ
1911
1912
1913
1901
ID=10
0 あいち
1 あおもり
n−2 おかやま
1902
ID=11
0 おきなわ

図19

【図２０】

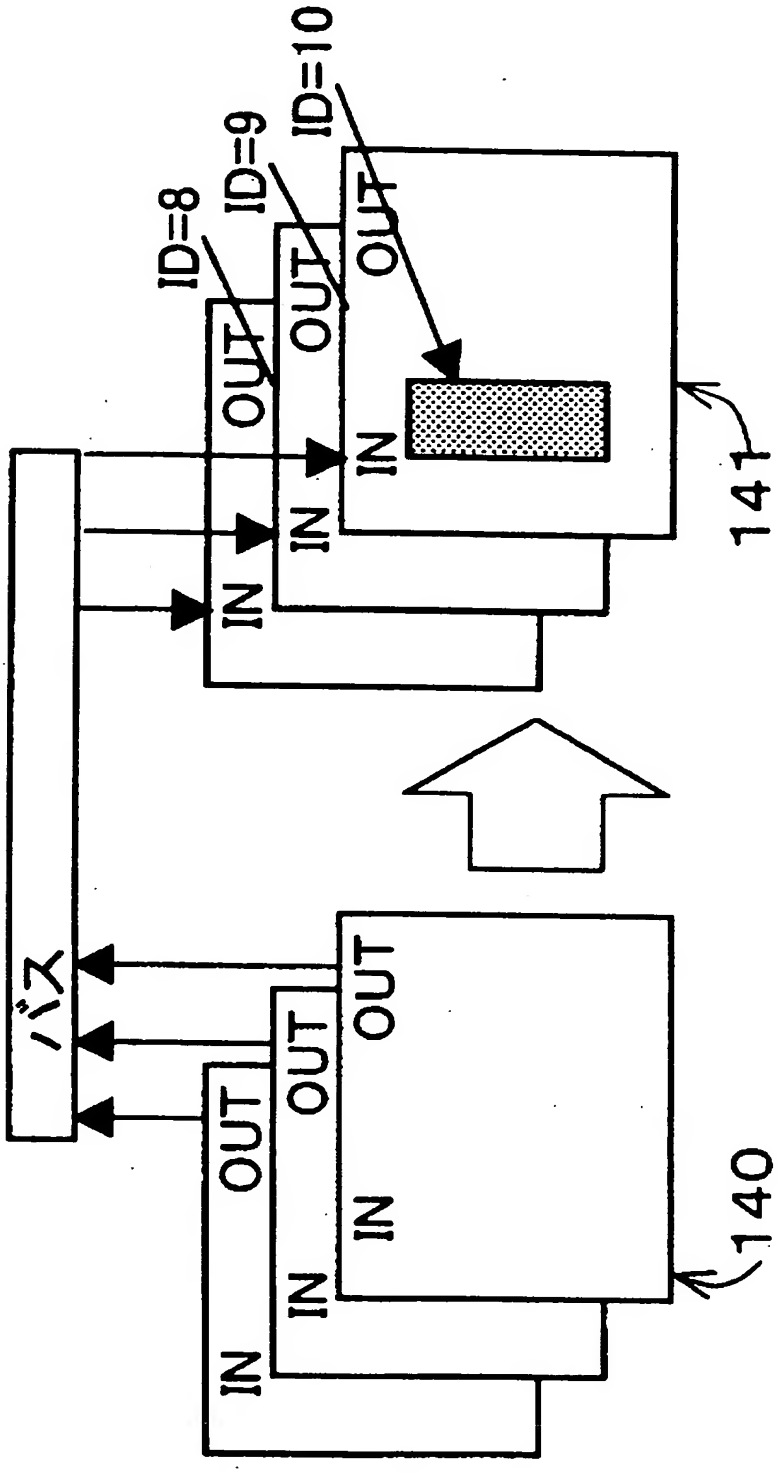


図20

【図２１】

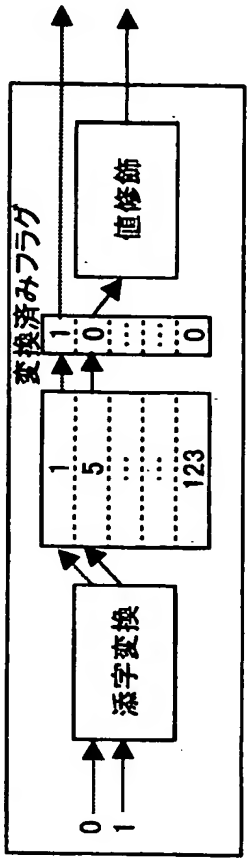


図21

【図２２】

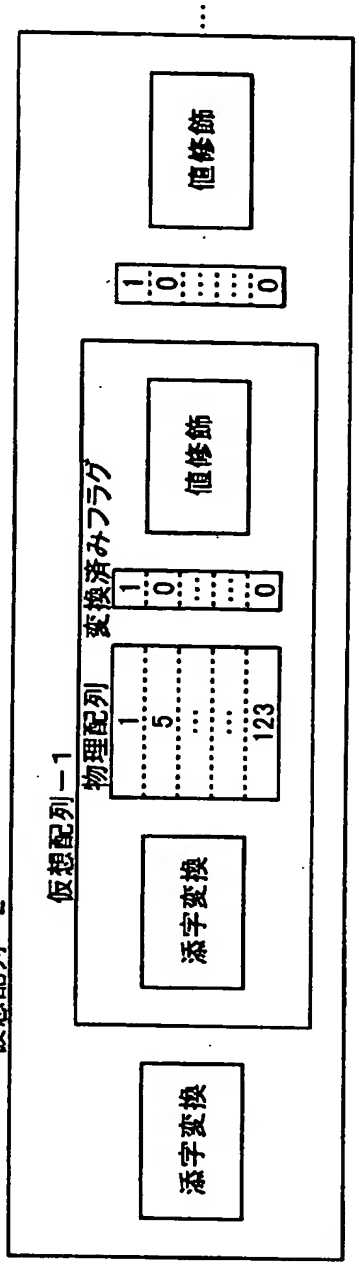


図22

【図２３】

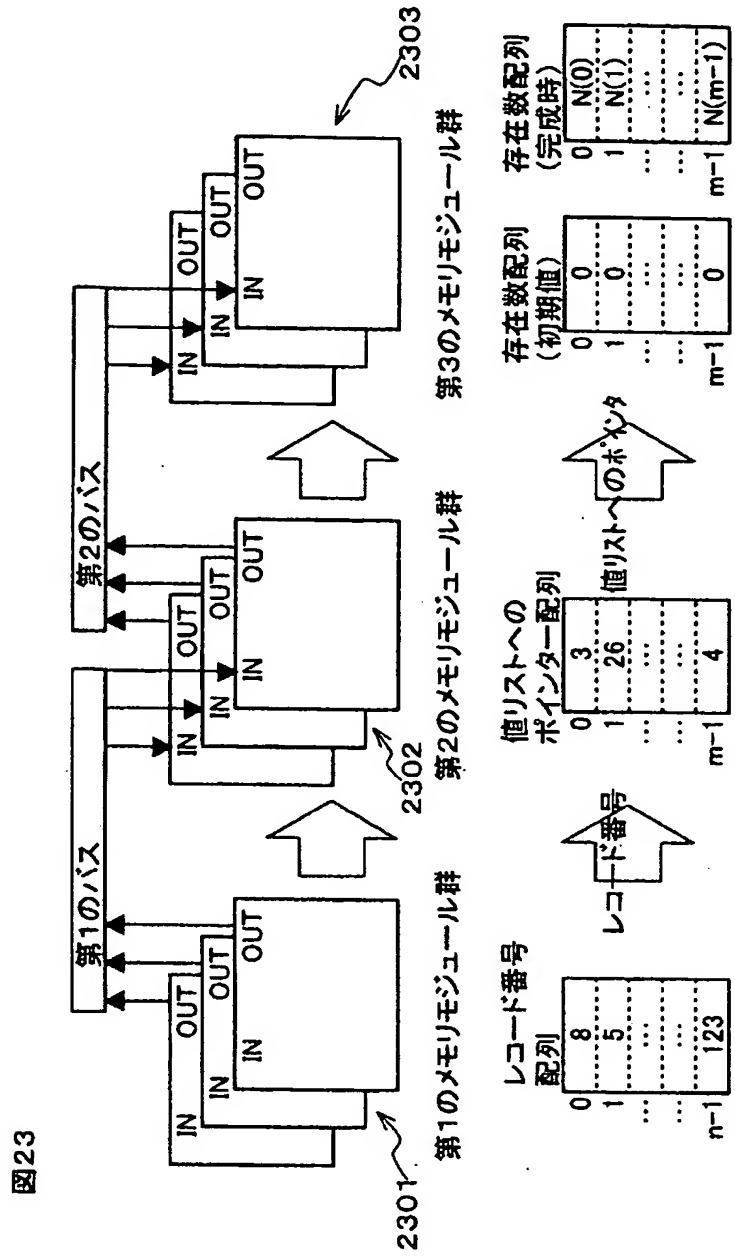

2303
第3のメモリモジュール群
第2のバス
IN
OUT
2302
第2のメモリモジュール群
第1のバス
2301
第1のメモリモジュール群
存在数配列
（完成時）
N(0)
N(1)
N(m-1)
存在数配列
（初期値）
0
値リストへのポインタ
値リストへの
ポインター配列
3
26
4
レコード番号
レコード番号
配列
8
5
123
n-1
m-1

図23

【図２４】

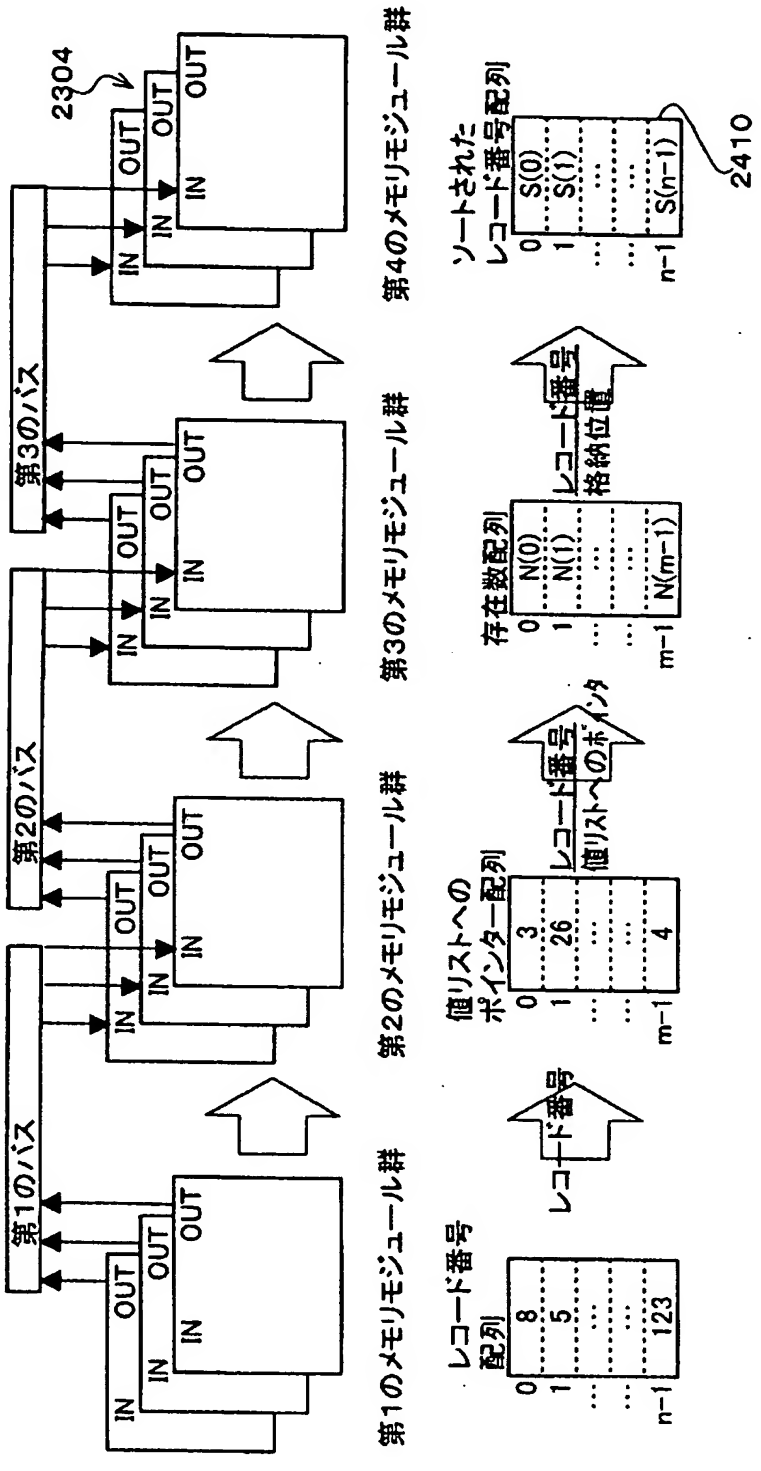
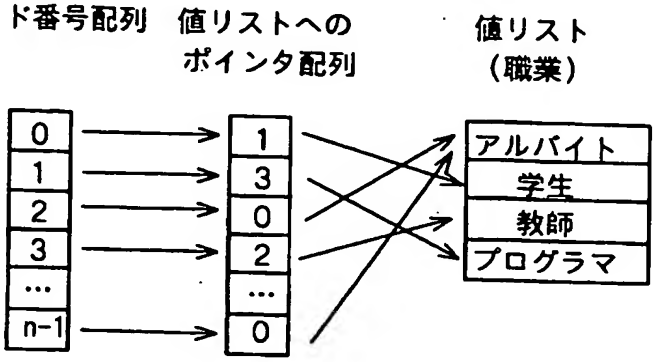

2304
第1のバス
第2のバス
第3のバス
IN
OUT
第1のメモリモジュール群
第2のメモリモジュール群
第3のメモリモジュール群
第4のメモリモジュール群
レコード番号
配列
0
1
n-1
8
5
123
レコード番号
値リストへの
ポインター配列
3
26
4
m-1
値リストへのポインタ
存在数配列
N(0)
N(1)
N(m-1)
レコード番号
格納位置
ソートされた
レコード番号配列
S(0)
S(1)
S(n-1)
2410

図24

【図２５】

図２５


レコード番号配列
値リストへの
ポインタ配列
値リスト
（職業）
0
1
2
3
…
n-1
1
3
0
2
…
0
アルバイト
学生
教師
プログラマ

【図２６】

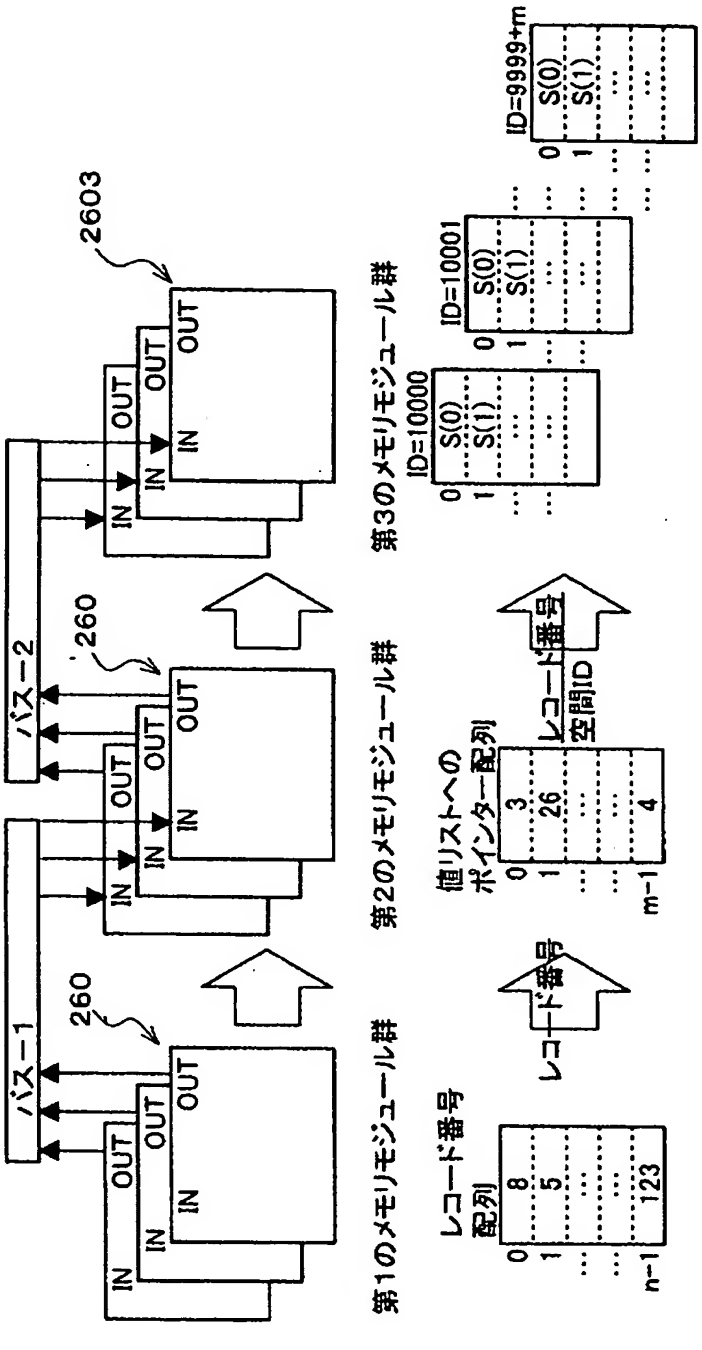

バス−1
バス−2
260
260
2603
IN
OUT
第1のメモリモジュール群
第2のメモリモジュール群
第3のメモリモジュール群
レコード番号
配列
値リストへの
ポインター配列
レコード番号
空間ID
ID=10000
ID=10001
ID=9999+m
S(0)
S(1)
0
1
n-1
m-1

図26

【図２７】

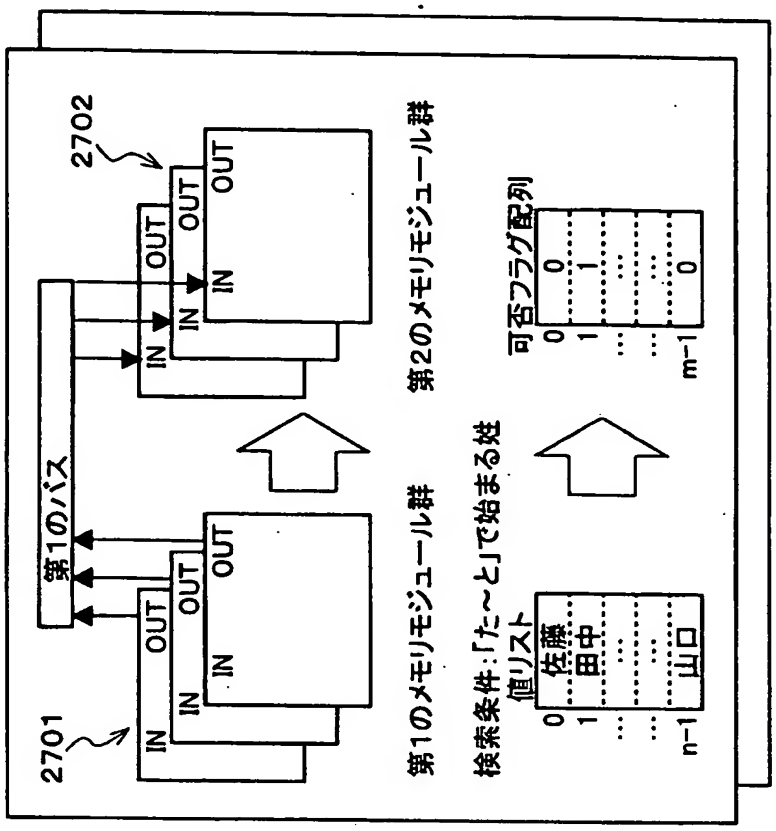

2702
2701
第1のバス
IN
OUT
第1のメモリモジュール群
第2のメモリモジュール群
検索条件：「た～と」で始まる姓
値リスト
佐藤
田中
山口
可否フラグ配列
0
1
n-1
m-1

図27

【図２８】

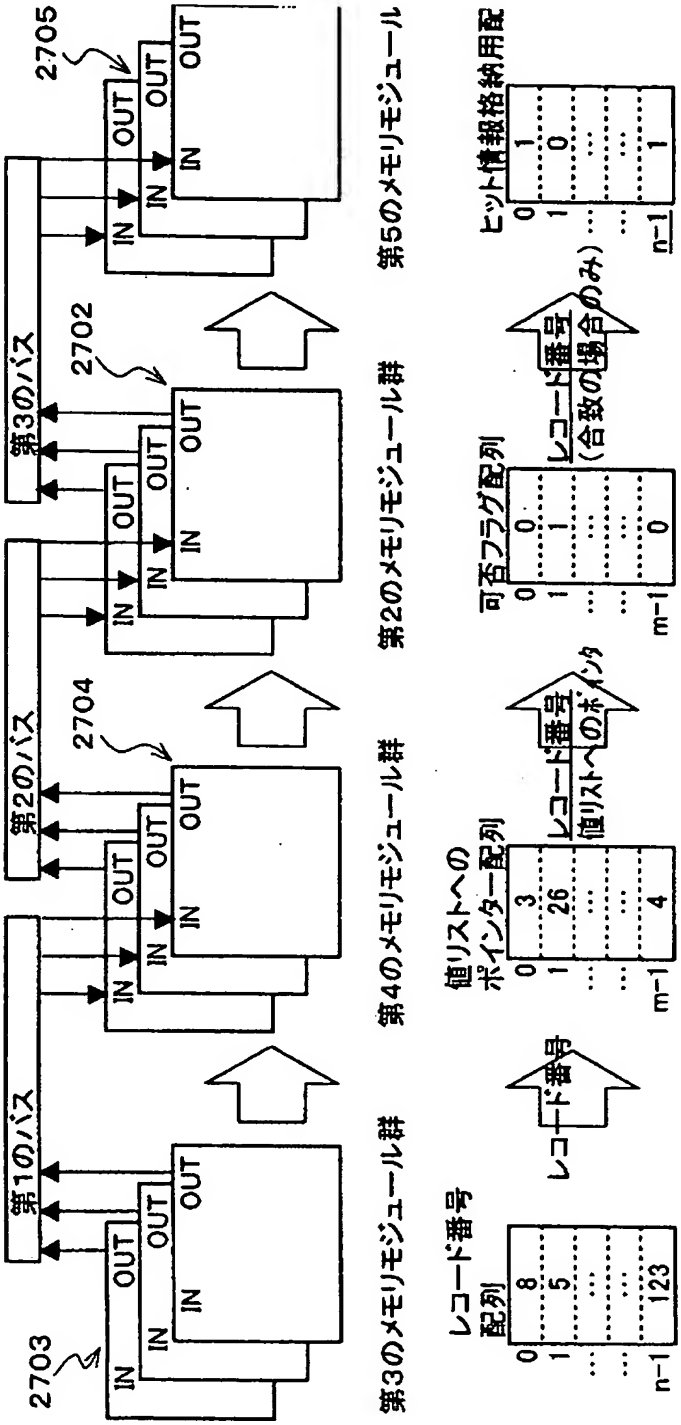

2705
2702
2704
2703
IN
OUT
第1のバス
第2のバス
第3のバス
第3のメモリモジュール群
第4のメモリモジュール群
第2のメモリモジュール群
第5のメモリモジュール
レコード番号
配列
値リストへの
ポインター配列
レコード番号
値リストへのポインタ
可否フラグ配列
レコード番号
(合致の場合のみ)
ヒット情報格納用配
0
1
n-1
m-1

図28

【書類名】　　要約書

【要約】

【課題】　分散メモリー型において、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供する

【解決手段】　コンピュータシステム１０は、ＣＰＵモジュール１２と、それぞれがＭＰＵ３６およびＲＡＭコア３４とを有する複数のメモリモジュール１４と、ＣＰＵとメモリモジュールとの接続やメモリモジュール間の接続をなす複数組のバス２４とを備え、ＣＰＵ１２から与えられるインストラクションにより、各メモリモジュールが作動する。所定の関連を有する一連のデータには、空間ＩＤが付与され、各メモリモジュールが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、一連のデータのサイズを含むテーブルを管理し、かつ、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータに関する処理を実行する。

【選択図】　図１